The Virgin Computer Games Series

Series Editor: Tim Hartnell

GAMES FOR YOUR APPLE IIe

£££££'s of entertaining games for only £2.95

Tony Dyson
Bjørn Engelhardt

Virgin

GAMES FOR YOUR APPLE IIe

By
Bjørn Engelhardt
and
Tony Dyson

GAMES FOR YOUR APPLE IIe

By
Bjørn Engelhardt
and
Tony Dyson

Virgin Books

First published in Great Britain in 1984 by Virgin Books Ltd, 61-63 Portobello Road, London W11 3DD.

ISBN 0 86369 046 7

Printed and bound in Great Britain by Richard Clay (The Chaucer Press) Ltd, Suffolk.

Production services by Book Production Consultants, Cambridge.

Designed by Ray Hyden.

Illustrated by Sue Walliker.

Typeset by QV Typesetting.

Distributed by Arrow Books.

TIM HARTNELL — THE SERIES EDITOR

Tim Hartnell is the most widely-published computer author in the world. Founder of the National ZX Users' Club, and founding editor of *ZX Computing* magazine, Tim has been involved over the years in a wide variety of computer activities. His published works include *The Personal Computer Guide* (Virgin Books) and *The Giant Book of Computer Games* (Fontana).

BJØRN ENGELHARDT AND TONY DYSON — THE AUTHORS

Bjørn and Tony are two teenage schoolboys from Lilydale, a suburb of Melbourne in Australia. They've both had more than two years' experience programming on Apple computers.

SUE WALLIKER — THE ILLUSTRATOR

Sue Walliker is a freelance illustrator.

CONTENTS

Editor's Introduction ... 13
Author's Introduction ... 15
Squash 'em ... 17
Target ... 19
Snake ... 23
Space Mission ... 26
Cannon ... 29
X-Lines ... 32
Treasure Nim ... 36
Duel ... 38
Nessie ... 41
Howzat ... 43
Simon Says ... 48
Golf ... 51
Phasar ... 54
Jackpot ... 57
Twenty One ... 61
Sonic Force ... 64
Morse Code ... 66
Hangman ... 69
Sequence Introduction ... 72
Doodle ... 75
How To Write Better Programs ... 79
Glossary ... 87
Bibliography ... 105

Editor's Introduction

Typing in a computer program is like opening an unknown door. You do not know until you actually open the door — or, in our case, run the program — what experience is waiting for you. Of course, the sign on the door has given you some indication, but nothing can equal first-hand experience.

You do not know precisely what experiences are waiting for you in the great programs in this book. Of course, if the introduction says you're entering a space game, it's very likely the program won't play 'Guess My Number' when you get it up and running. But the listing rarely hints at the computer's game-playing strategy, or the screen display, or the fun that is waiting for you.

This book has a number of unknown doors — doors leading into outer space and into the fiendish worlds of computer intelligence, wizards and Adventure.

We've provided the doors...and the keys. All you have to do to turn the lock is type in the program, and run it. Whatever you find behind each door, I guarantee you won't be disappointed.

Tim Hartnell
Series editor
London
March 1984

Authors' Introduction

Apple computers have been around a long time, yet there always seem to be new discoveries which can be made about them. There are hundreds of games for the Apple, but still people are interested in new games — games which will provide them with more fun than other games they've played, and games which will show them new programming techniques to apply to their own work.

Over the past two years, we've had a lot of fun writing programs for the Apple, and in this book we've brought the results of those two years to you. We hope you will have as much fun playing these games as we have had writing them, and that you'll pick up a few ideas for your own programming and games writing.

Whether you want brain-stretching entertainment, or you're more interested in 'shoot-em-ups', you'll find your interests have been catered for.

Have fun!

Bjørn Engelhardt and Tony Dyson
Lilydale, Victoria, Australia
May 1984

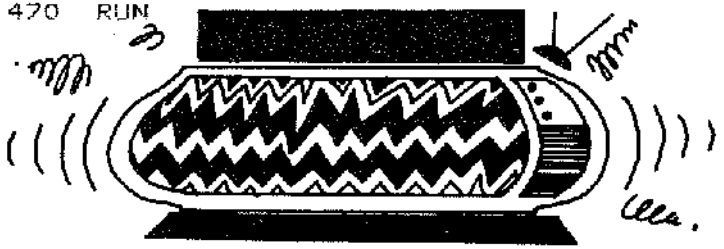
SQUASH 'EM

The object of this game is to survive for as long as possible without being squashed. There is only one safe spot, and that is on the very left hand side of the screen. However, if you stay here, your score does not increase.

You control your man with the right and left arrow keys. Touching any other key will cause you to stop. We challenge you to beat our high score of 253...

```
10  REM   SQUASH EM
20  REM   B.ENGELHARDT
30  POKE  - 16368,0
40  GR : HOME
50  VTAB 22: INVERSE : PRINT "SCORE:": N
ORMAL
60  GOSUB 120
70  GOSUB 160
80 ST = ST + 1: IF ST > 4 THEN  GOSUB 23
0
90  GOSUB 390
100  GOSUB 440
110  GOTO 70
120  FOR A = 1 TO 20: COLOR= A
130  HLIN 0,39 AT A: NEXT
140  HLIN 0,39 AT 39
150  RETURN
160 K =  PEEK ( - 16384)
170  IF K = 136 THEN MM = MM - 2
180  IF K = 149 THEN MM = MM + 2
190  IF MM < 0 THEN MM = 39
200  IF MM > 39 THEN MM = 0
210  COLOR= 9: VLIN 35,38 AT MM
220  RETURN
230 R1 =  INT ( RND (1) * 10) + 1
240 ST = 0
```

```
250  IF MM > 25 THEN R1 = R1 + 26: GOTO
280
260  IF MM > 12 THEN R1 = R1 + 14: GOTO
280
270  IF MM > 10 THEN R1 = R1 + 10
280  COLOR= 4: FOR SQ = 21 TO 29: HLIN R
1,R1 + 3 AT SQ: NEXT
290  GOSUB 440: GOSUB 160
300  COLOR= 4
310  FOR SQ = 30 TO 38: HLIN R1,R1 + 3 A
T SQ: NEXT
320  GOSUB 440: GOSUB 160
330  IF  SCRN( MM,32) <  > 0 OR  SCRN( M
M,31) <  > 0 THEN 460
340  COLOR= 0: FOR SD = 38 TO 29 STEP  -
 1: HLIN R1,R1 + 3 AT SD: NEXT
350  GOSUB 440: GOSUB 160
360  COLOR= 0
370  FOR SD = 28 TO 21 STEP  - 1: HLIN R
1,R1 + 3 AT SD: NEXT : GOSUB 440:
      GOSUB 160
380  RETURN
390  VTAB 22: HTAB 7
400  IF MM = 0 OR MM = 39 THEN  RETURN
410  PRINT TS
420 TS = TS + 1
430  RETURN
440  COLOR= 0: VLIN 35,38 AT MM
450  RETURN
460  FOR A = 1 TO 50: POKE  - 16303,0:S
=  PEEK ( - 16336): POKE  - 16304
     ,0: NEXT
470  RUN
```



TARGET

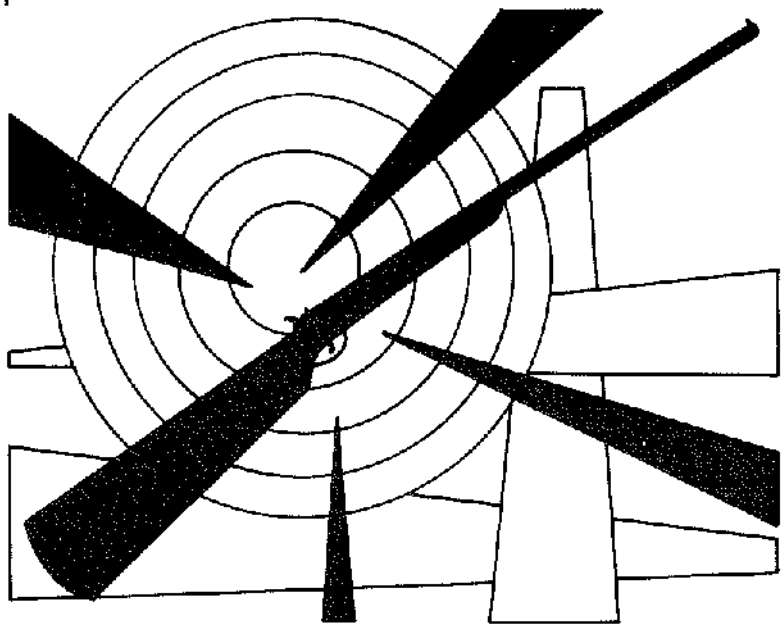
This game will really test your skill and accuracy at shooting. As the target bounces around the screen, you line up your cross-hairs with it and pull the trigger! You have 10 shots; every time you hit the target there is a colourful explosion, and you get one point.

Your cross-hairs are controlled by the keys:

```
  I
J + K
  M
```

The 'trigger' is the space bar.

Every now and then, the target will randomly change its direction and speed, so beware. Good luck, and happy shooting!

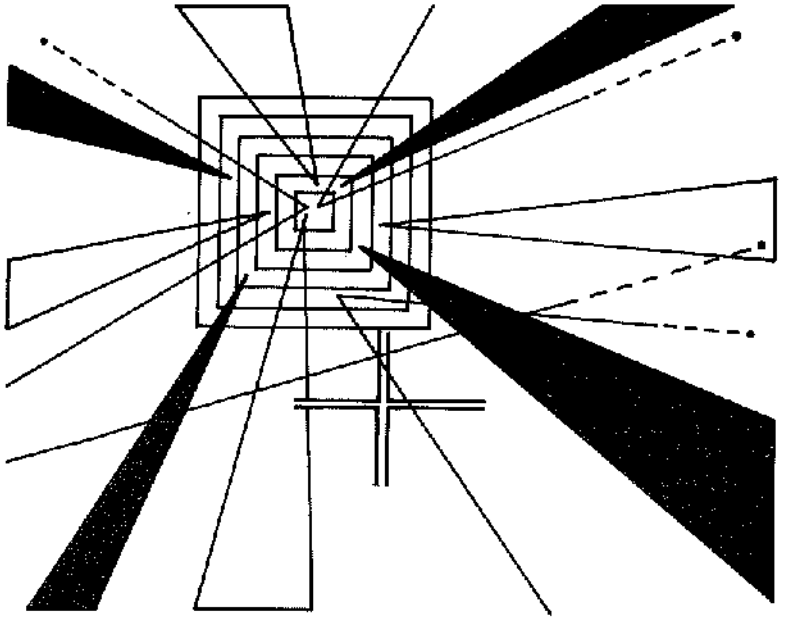


```
10  REM  TARGET
20  REM  TONY DYSON '84
30  GOSUB 100: REM  INITIALIZE
40  GOSUB 230: REM  DRAWING CYCLE
50  GOSUB 330: REM  GET INPUT
60  IF F = 1 THEN F = 0: GOSUB 380: REM
 FIRE
70  GOSUB 480: REM  MOVE OBJECTS
80  GOTO 40
90  REM  INITIALIZE
100  GR : HOME
110 SX = 20:SY = 20:S = 0:B = 10:XS = SX
:YS = SY
120  VTAB 23: INPUT "TARGET SIZE (1-4):"
;L
130  IF L < 0 OR L > 4 THEN  VTAB 23: CA
LL  - 868: GOTO 120
140  VTAB 23: CALL  - 868: INPUT "LEVEL
(1-5):";G
150  IF G < 1 OR G > 5 THEN 200
160  DEF  FN R(S) =  INT ( RND (S) * (40
 - L)) + 1
170  DEF  FN O(S) =  INT (( RND (S) * G
* 4) + 1) / 4
180 X =  FN R(1):Y =  FN R(1):XC = X:YC
= Y
190 XO =  FN O(1):YO =  FN O(1)
200 T =  INT ( RND (1) * 15) + 1
210  RETURN
220  REM  DRAWING CYCLE
230  COLOR= 0: FOR I = 0 TO L - 1: HLIN
XC,XC + L AT YC + I
240  NEXT :XC = X:YC = Y
250  COLOR= T: FOR I = 0 TO L - 1: HLIN
X,X + L AT Y + I: NEXT
260  COLOR= 0: VLIN YS - 4,YS + 4 AT XS
270  HLIN XS - 2,XS + 2 AT YS:XS = SX:YS
 = SY
280  COLOR= 13: VLIN SY - 4,SY + 4 AT SX
: HLIN SX - 2,SX + 2 AT SY
```



```
290  VTAB 22: CALL  - 868: PRINT "SCORE:
";S: VTAB 23: CALL  - 868
300  PRINT "SHELLS LEFT:";B
310  RETURN
320  REM  GET INPUT
330 KI =  PEEK ( - 16384) - 200
340  IF KI = 5 THEN KI = 4
350  IF KI =  - 40 THEN F = 1
360  RETURN
370  REM  FIRE
380  COLOR= 1: VLIN SY - 4,SY + 4 AT SX:
 HLIN SX - 2,SX + 2 AT SY
390  FOR I = 1 TO 10:SND =  PEEK ( - 163
36): NEXT
400  IF SY >  = Y AND SY <  = Y + L AND
SX >  = X AND SX <  = X + L THEN
     430
410 CK =  PEEK ( - 16368):B = B - 1: IF
B = 0 THEN 630
420  RETURN
430  FOR I = 1 TO 25: POKE  - 16299,0: P
OKE  - 16300,0
440 SND =  PEEK ( - 16336): NEXT
450 X =  FN R(1):Y =  FN R(1):XO =  FN O
(1)
460 YO =  FN O(1):S = S + 1: GOTO 410
470  REM  MOVE TARGET/SIGHTS
480  IF KI < 1 OR KI > 4 THEN 570
490  ON KI GOTO 500,520,540,560
500  IF SY - 4 > 0 THEN SY = SY - 1
510  GOTO 570
520  IF SX - 2 > 0 THEN SX = SX - 1
530  GOTO 570
540  IF SX + 2 < 39 THEN SX = SX + 1
550  GOTO 570
560  IF SY + 4 < 39 THEN SY = SY + 1
570  IF X + XO < 0 OR X + L + XO > 38 TH
EN XO =  - XO:R = 1
580  IF Y + YO < 0 OR Y + L + YO > 38 TH
EN YO =  - YO:R = 1
```

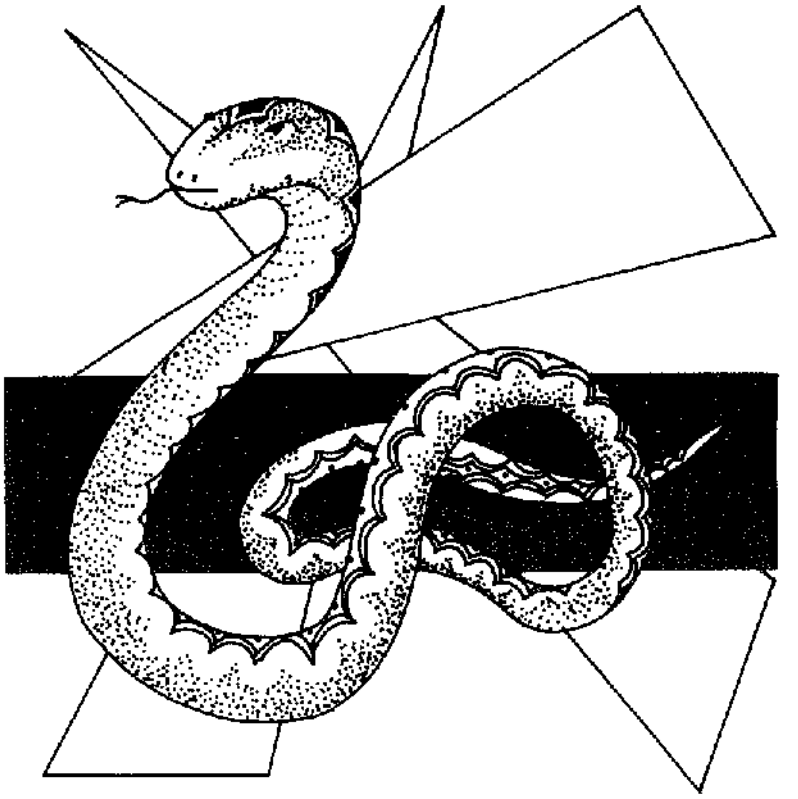
```
590  IF R = 1 THEN R = 0: GOTO 610
600  IF  RND (1) > 0.96 THEN XO =  FN O(
1):YO =  FN O(1)
610 X = X + XO:Y = Y + YO
620  RETURN
630  TEXT : HOME
640  PRINT : PRINT "YOU HAVE RUN OUT OF
SHELLS!"
650  PRINT : PRINT "YOUR RATING WAS:": P
RINT : PRINT S * G / L
660  PRINT : INPUT "TRY AGAIN? ";YN$: IF
 YN$ = "Y" THEN  RUN
670  END
```



SNAKE

This is one of the most addictive games in the book. You control a snake that constantly changes colour. You must manoeuvre it through randomly deposited mines. If you hit a mine or run into yourself you are instantly destroyed.

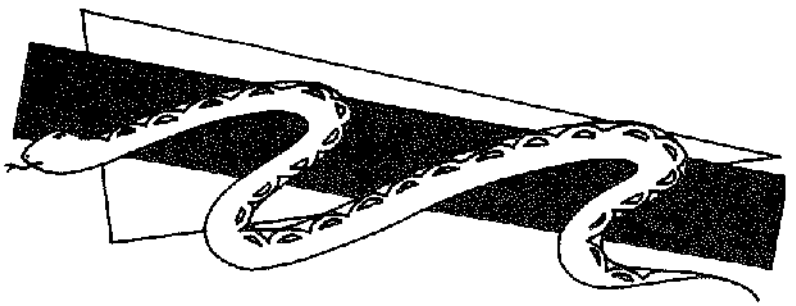
The keys are A to move up, Z to move down, the right arrow to move right, and the left arrow to move left.



```
10  REM  *** SNAKE
20  REM  *** B.ENGELHARDT
30  HOME : VTAB 22: PRINT "LEVELS 1 - 10
"
40  VTAB 24: PRINT "YOUR CHOICE =>";: IN
PUT C$:C =  VAL (C$)
50  IF C < 1 OR C > 10 THEN 560
60  HOME : VTAB 22: PRINT "SPEED 1 - 10"
70  VTAB 24: INPUT "YOUR CHOICE =>";SP
80  IF SP < 1 OR SP > 10 THEN 560
90 DE = ((SP - 10) * 15)
100 DE =  ABS (DE)
110 RN =  INT (12 * C)
120 SEC = 0: GR : FOR BL = 1 TO RN
130  COLOR=  INT (12 *  RND (1)) + 3
140  PLOT  INT (39 *  RND (1)) + 1, INT
(39 *  RND (1)) + 1
150  NEXT
160  COLOR= 12
170  HLIN 0,39 AT 0: HLIN 0,39 AT 39: VL
IN 1,38 AT 0: VLIN 0,38 AT 39
180 HI = 0
190 C = 2
200 X = 20:Y = 2
210  POKE  - 16368,0
220 K =  PEEK ( - 16384)
230  IF K = 193 THEN 290
240  IF K = 218 THEN 320
250  IF K = 149 THEN 350
260  IF K = 136 THEN 380
270 CS = 1
280  GOTO 410
290  IF  SCRN( X,Y - 1) > 0 THEN HI = 1
300 Y = Y - 1
310  GOTO 410
320  IF  SCRN( X,Y + 1) > 0 THEN HI = 1
330 Y = Y + 1
340  GOTO 410
350  IF  SCRN( X + 1,Y) > 0 THEN HI = 1
360 X = X + 1
```



```
370  GOTO 410
380  IF  SCRN( X - 1,Y) > 0 THEN HI = 1
390 X = X - 1
400  GOTO 410
410 CO = CO + 1: IF CO > 14 THEN CO = 2
420  COLOR= CO
430  PLOT X,Y
440  IF CS = 1 THEN 460
450 SEC = SEC + 1
460  IF HI = 1 THEN 500
470  FOR TI = 1 TO DE: NEXT
480 CS = 0
490  GOTO 220
500  FOR A = 1 TO 50: POKE  - 16303,0: P
OKE  - 16304,0: NEXT :
510  HOME : VTAB 24: INVERSE : PRINT "SC
ORE:";: NORMAL : PRINT  INT (SEC /
     7.2);: HTAB 20: INVERSE : PRINT "HI
-SCORE:";: NORMAL
520  IF  INT (SEC / 7.2) > HS THEN HS =
 INT (SEC / 7.2)
530  PRINT HS
540  FOR TI = 1 TO 2000: NEXT
550  GOTO 30
560  TEXT
570 DE = 0
580  HOME : INVERSE : VTAB 12: HTAB 17:
FLASH : PRINT "SUFFER": FOR TI =
     1 TO 2000: NEXT :RN = 200: GOTO 120
```

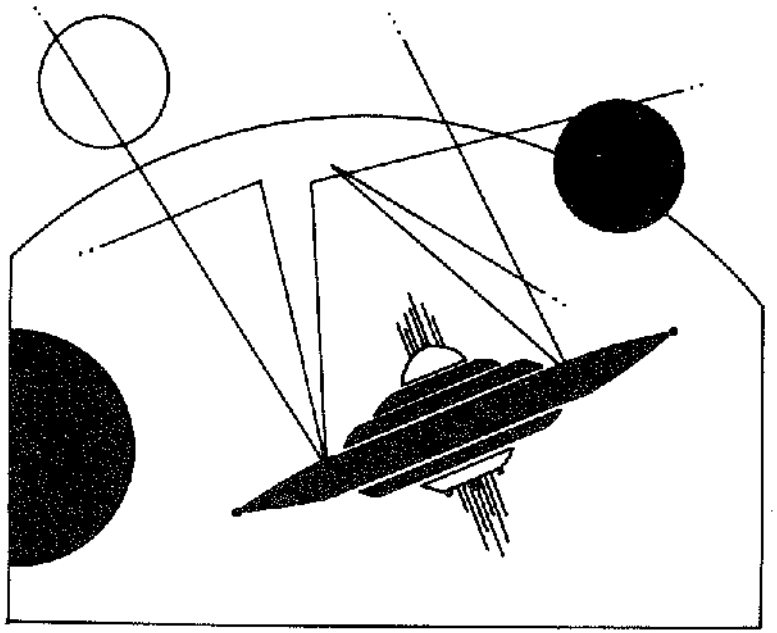


SPACE MISSION

You are the commander of a large fleet of space cruisers. You have been alerted to the presence of a large unidentified object in your section of space, but you can't pick it up on your radar.

You can, however, pick up the explosion of a plasma torpedo on your radar, as well as the reflection of the explosion off the object as well, though not for long enough for it to appear on the screen.

Each time you fire, your battle computer tells you in which direction the shot was wide. For every shot, you must enter three co-ordinates — the X co-ordinate (east-west), the Y co-ordinate (north-south), and the distance.




```
10  REM  SPACE MISSION
20  REM  TONY DYSON '83
30  TEXT : HOME :S = 1: DIM P(3),S(3)
40 D =  INT ( RND (1) * 40) + 1
50 T =  INT (6 * D / ( RND (1) * 20 + 1)
)
60  FOR I = 1 TO 3:P(I) =  INT ( RND (1)
 * D) + 1: NEXT
70  VTAB 7: PRINT "YOUR ATTACK FORMATION
 IS ";D;" BY ";D;" SHIPS."
71  PRINT "YOUR TORPS HAVE A RANGE OF ";
D;" WARPS."
80  PRINT : PRINT "YOU HAVE ";T;" ENERGY
 TORPEDOES LEFT."
81  PRINT : PRINT "GOOD LUCK..."
90  GOSUB 330: PRINT : PRINT : PRINT  TA
B( 10);"< HIT ANY KEY >": GET CN$

100  HOME : GOTO 120
110  HOME : PRINT "X COORD.";S(2): PRINT
 "Y COORD.";S(1)
111  PRINT "BOOST ";S(3): PRINT : PRINT
A$
120 A$ = "": VTAB 7: PRINT T - S;" TORPE
DOES LEFT."
130  VTAB 10: CALL  - 868: PRINT "SHOT N
O.";S
140  VTAB 12: CALL  - 868: INPUT "INPUT
CO-ORDINATES (X,Y,DIST.):";S(2),S
     (1),S(3)
150  FOR I = 1 TO 3: IF S(I) < 1 OR S(I)
 > D THEN 140: NEXT
160  IF S(1) = P(1) AND S(2) = P(2) THEN
 A$ = "X,Y O.K.": GOTO 220
170  IF S(1) < P(1) THEN A$ = "NORTH": G
OTO 190
180  IF S(1) > P(1) THEN A$ = "SOUTH"
190  IF S(2) < P(2) THEN A$ = A$ + "WEST
": GOTO 210
```

```
200  IF S(2) > P(2) THEN A$ = A$ + "EAST
"
210 A$ = "SHOT " + A$ + " OF TARGET.": G
OTO 230
220  IF S(3) = P(3) THEN A$ = "YOU DESTR
OYED THE TARGET!": GOTO 300
230  IF S(3) = P(3) THEN A$ = A$ + "DIST
ANCE O.K.": GOTO 260
240  IF S(3) < P(3) THEN A$ = A$ +  CHR$
 (13) + "BOOST TOO WEAK.": GOTO 2
     60
250 A$ = A$ + "BOOST TOO STRONG."
260 S = S + 1: IF S <  = T THEN 110
270  VTAB 20: PRINT "YOU HAVE RUN OUT OF
 TORPEDOES!"
280  PRINT "THE ENEMY CRAFT WAS AT ";P(1
);",";P(2);",";P(3);"."
290  PRINT "BETTER LUCK NEXT TIME...": G
OTO 320
300  VTAB 14: CALL  - 868: PRINT A$
310  PRINT : PRINT "YOU USED ";S;" TORPE
DOES."
320  END
330  FOR I = 771 TO 789: READ J: POKE I,
J: NEXT
340  FOR I = 1 TO 17: READ J,K: POKE 0,2
55 - J: POKE 1,K: CALL 771: NEXT

350  RETURN
360  DATA  173,48,192,136,208,4,198,1,24
0,8,202,208,246,166,0,76,3,3,96
370  DATA  63,240,94,50,84,50,63,50,84,2
40,111,50,94,50,84,50,94,240
380  DATA  127,50,111,50,94,50,84,50,111
,50,94,50,84,50,63,250
```


CANNON

In this game you control a cannon on the left of the screen and your aim is to hit the castle on the right five times. If you accumulate five misses you lose.

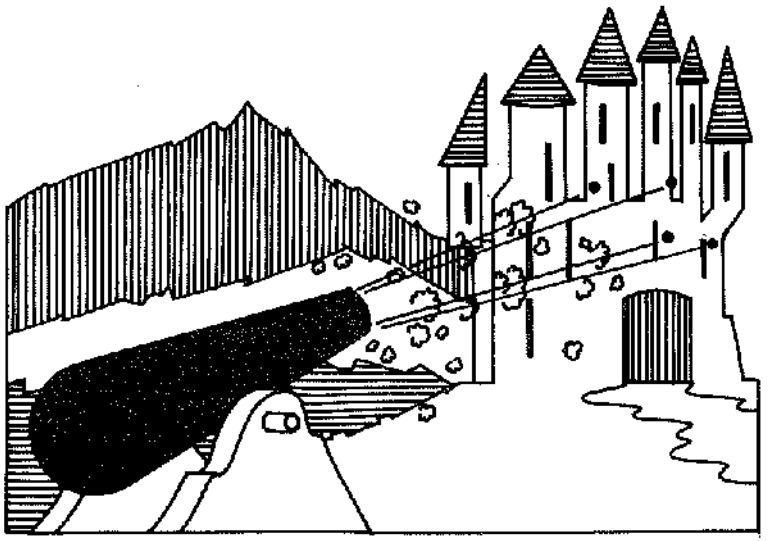
An imaginary wind gusts at random, so take care. You control the cannon by using A to move up, Z to move down, and the space bar to fire.

```
10  REM  *** CANNON
20  REM  *** B.ENGELHARDT
30  GOSUB 550: REM  *** SHAPE
40  HOME
50  HGR
60  HCOLOR= 3: HPLOT 0,0: CALL 62454: HC
OLOR= 0
70  FOR A = 100 TO 115: HPLOT 3,A TO 10,
A: HPLOT 17,A TO 24,A
80  HPLOT 31,A TO 38,A: NEXT
90  FOR A = 115 TO 122: HPLOT 250,A TO 2
57,A: HPLOT 264,A TO 271,A
100  HPLOT 277,A TO 279,A: NEXT
110  FOR A = 122 TO 140: HPLOT 250,A TO
279,A: NEXT
120  FOR A = 115 TO 140: HPLOT 3,A TO 38
,A: NEXT
130 Y1 = 104
140  GOSUB 260
150  GOSUB 300
160  HOME : VTAB 24: PRINT "HITS:";HI;:
HTAB 20: PRINT "MISSES:";SF
170  SCALE= 1: ROT= 1
180  GET K$
190  IF K$ = "A" THEN  GOSUB 230:Y1 = Y1
 - 1: GOSUB 260
```

```
200  IF K$ = "Z" THEN  GOSUB 230:Y1 = Y1
 + 1: GOSUB 260
210  IF K$ = " " THEN  GOTO 320
220  GOTO 180
230  HCOLOR= 3: FOR ER = 104 TO 111: HPL
OT 38,ER TO 50,Y1 + ER - 104
240  NEXT
250  RETURN
260  HCOLOR= 5: FOR DR = 104 TO 111: HPL
OT 38,DR TO 50,Y1 + DR - 104
270  NEXT
280  RETURN
290  GOTO 180
300 X = 53
310  RETURN
320 G2 = (((105 - (105 - Y1)) / 105) - 1
)
330 G = G2 * (( INT (2 *  RND (1)) + 1)
+ 12)
340 X1 = 1.7 + ( INT (9 *  RND (1)) / 3)
350 Y = Y1
360 G1 = 0.04
370  SCALE= 1: ROT= 0
380 X = X + X1:Y = Y + G: HCOLOR= 2: DRA
W 1 AT X,Y: HCOLOR= 3
390  DRAW 1 AT X,Y:G = G + G1
400  IF X > 250 AND Y > 115 THEN 440
410  IF X + X1 > 279 OR Y > 135 THEN 500
420  GOTO 370
430  GOTO 150
440  HCOLOR= 3
450  VTAB 24
460 HI = HI + 1
470  IF HI =  > 5 THEN 610
480  FOR A = 1 TO 10: SCALE= A / 3 + 0.7
: ROT= A
490  DRAW 1 AT X,Y: NEXT : GOTO 150
500  HCOLOR= 4: FOR A = 1 TO 10: SCALE=
A / 3 + 0.7: ROT= A: DRAW 1 AT X,
     Y
```



```
510  NEXT : HCOLOR= 3: FOR A = 1 TO 10:
SCALE= A / 3 + 0.7: ROT= A
520  DRAW 1 AT X,Y: NEXT
530 SF = SF + 1: IF SF >  = 5 THEN 630
540  GOTO 150
550  FOR PO = 24576 TO 24593
560  READ Z
570  POKE PO,Z
580  NEXT
590  POKE 232,0: POKE 233,96
600  RETURN
610  HOME : VTAB 22: INVERSE : HTAB 16:
PRINT "YOU WIN": NORMAL
620  FOR A = 1 TO 2000: NEXT : GOTO 650
630  HOME : VTAB 22: HTAB 15: INVERSE :
PRINT "YOU LOOSE": NORMAL
640  FOR A = 1 TO 2000: NEXT
650  FOR A = 1 TO 6: HCOLOR= A: HPLOT 0,
0: CALL 62454: NEXT : RUN
660  DATA  1,0,4,0,45,46,63,63,62,45,45,
45,55,63,63,53,45,0,0,0
```



X-LINES

The best way to get the idea of this game is to type in the program 'X-pic' and watch it run for a few minutes before reading this introduction. When you have done this, you will notice that the computer divides up the screen into four sections using two intersecting pink lines. Each section has a different colour, and the spot where the lines cross is yellow.

The 'X-lines' program is basically the same, except that you can't see the divisions on the screen. You have to try and find the yellow spot, in the least amount of guesses possible. You do this by guessing points on the screen which will then be displayed.

To move the two brown 'cursor lines', use the keys I, J, K and M, which move up, left, right and down respectively. When you have the cursors lined up with the spot you want press RETURN. The point you indicated will then be displayed, and the cursors moved back to 'home' position.

To save time, I suggest you use the (REPT) key while moving the cursors.

```
10  REM  X-PIC
20  REM  TONY DYSON '84
30  REM  DISPLAY FOR X-LINES
40  GR : HOME
50  DIM S(40,40),D(40,40)
60  DEF  FN A(X) = M * X + B
70  DEF  FN B(X) = N * X + C
80 A1 =  INT ( RND (1) * 181): IF A1 = 9
0 THEN 290
90 A1 = A1 * 20.1416 / 180:M =  TAN (A1)
100 B =  INT ( RND (1) * 80)
110 XC =  INT ( RND (1) * 80) + 1
120  IF  FN A(XC) < 1 OR  FN A(XC) > 40
THEN  RUN
```



```
130 A2 =  INT ( RND (1) * 181): IF A2 =
90 THEN 130
140 A2 = A2 * 20.1416 / 180:N =  TAN (A2
)
150 C = XC * (M - N) + B
160  FOR X = 0 TO 39:L =  INT ( FN A(X))
170  FOR Y = 0 TO 39:T =  SGN (L - Y): C
OLOR= T + 3: PLOT X,Y
180  NEXT :SND =  PEEK ( - 16336): NEXT
190  FOR X = 0 TO 39:L =  INT ( FN B(X))
: IF L < 0 THEN 270
200  IF L > 39 THEN  FOR Y = 0 TO 39: GO
TO 220
210  FOR Y = 0 TO L
220  COLOR=  SCRN( X,Y) + 10
230  IF Y = L AND X <  > XC THEN  COLOR=
 3
240  IF Y =  INT ( FN A(X)) AND X <  > X
C THEN  COLOR= 3
250  PLOT X,Y
260  NEXT
270 SND =  PEEK ( - 16336): NEXT
280  GOTO 80
290 XC =  INT ( RND (1) * 80) + 1
300  FOR Y = 1 TO 40:S(XC,Y) = 2: NEXT
310 A2 =  INT ( RND (1) * 71): IF  RND (
1) < 0.5 THEN A2 = 180 - A2
320 YC =  INT ( RND (1) * 80) + 1:B = YC
330  FOR X = 1 TO 40:L =  INT ( FN A(X))
340  IF X = XC THEN 360
350  FOR Y = 1 TO 40:T =  SGN (L - Y):S(
X,Y) = T + 2: NEXT
360  NEXT : GOTO 220
370 I =  PEEK ( - 16384):I = I - 200
380  PRINT I: GOTO 370
```

```
10  REM  X-LINES
20  REM  TONY DYSON '84
30  DIM S(40,40),D(40,40): GR : HOME
40  DEF  FN A(X) = M * X + B
50  DEF  FN B(X) = N * X + C
60 A1 =  INT ( RND (1) * 181): IF A1 = 9
0 THEN 450
70 A1 = A1 * 20.1416 / 180:M =  TAN (A1)
80 B =  INT ( RND (1) * 60)
90 XC =  INT ( RND (1) * 60) + 1
100  IF  FN A(XC) < 1 OR  FN A(XC) > 40
THEN  RUN
110 A2 =  INT ( RND (1) * 181): IF A2 =
90 THEN 110
120 A2 = A2 * 20.1416 / 180:N =  TAN (A2
)
130 C = XC * (M - N) + B
140  FOR X = 0 TO 39:L =  INT ( FN A(X))
150  FOR Y = 0 TO 39:T =  SGN (L - Y):S(
X + 1,Y + 1) = T + 3
160  NEXT : NEXT
170  FOR X = 0 TO 39:L =  INT ( FN B(X))
: IF L < 0 THEN 240
180  IF L > 39 THEN  FOR Y = 0 TO 39: GO
TO 200
190  FOR Y = 0 TO L
200 S(X + 1,Y + 1) = S(X + 1,Y + 1) + 10
210  IF Y = L AND X <  > XC THEN S(X + 1
,Y + 1) = 3
220  IF Y =  INT ( FN A(X)) AND X <  > X
C THEN  COLOR= 3
230  NEXT
240  NEXT
250  GOSUB 350:X = X - 1:Y = Y - 1
260 D(X + 1,Y + 1) = S(X + 1,Y + 1): COL
OR= D(X + 1,Y + 1)
270  PLOT X,Y
280  IF D(X + 1,Y + 1) <  > 13 THEN G =
G + 1: GOTO 250
290  PRINT "YOU'RE SPOT ON!"
```

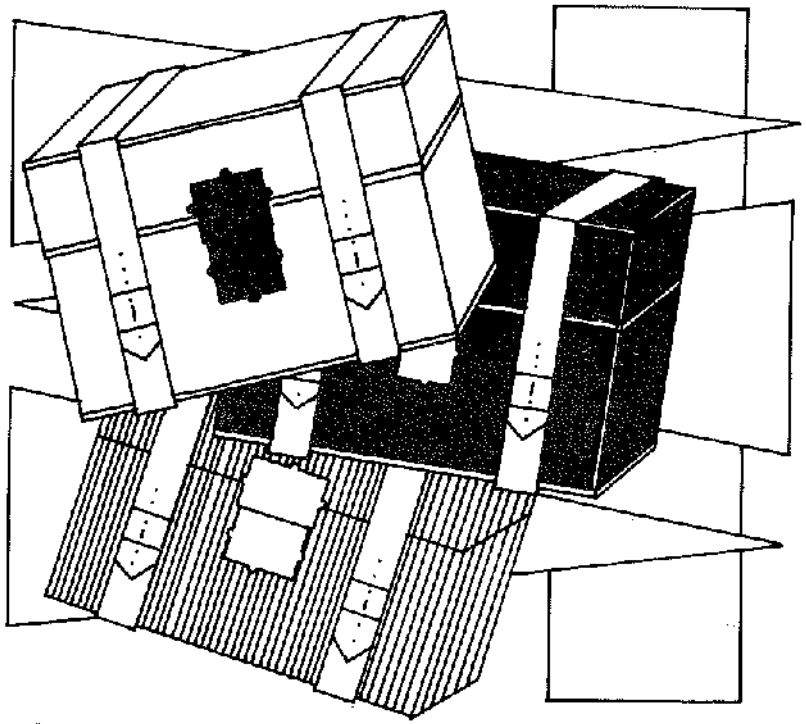


```
300  PRINT "IT TOOK YOU ";G;" GUESSES."
310  PRINT "ANOTHER GAME?";
320  GET YN$: IF YN$ <  > "Y" AND YN$ <
 > "N" THEN 320
330  IF YN$ = "Y" THEN  RUN
340  END
350 X = 20:Y = 20:X1 = X:Y1 = Y: GOTO 44
0
360 I =  PEEK ( - 16384):CS =  PEEK ( -
16368): IF I < 128 THEN 360
370 I = I - 200: IF I = 5 THEN I = 4
380  IF I =  - 59 THEN RT = 1
390  IF (I + 2) / 3 =  INT ((I + 2) / 3)
 THEN Y1 = Y +  SGN (I - 2): GOTO
     410
400 X1 = X +  SGN (I * 2 - 5): IF X1 <
1 OR X1 > 40 THEN X1 = X
410  FOR RY = 1 TO Y: COLOR= D(X,RY): PL
OT X - 1,RY - 1: NEXT
420  FOR RX = 1 TO X: COLOR= D(RX,Y): PL
OT RX - 1,Y - 1: NEXT
430  IF RT = 1 THEN RT = 0: RETURN
440 X = X1:Y = Y1: COLOR= 8: VLIN 0,Y -
1 AT X - 1: HLIN 0,X - 1 AT Y - 1
     : GOTO 360
450 XC =  INT ( RND (1) * 60) + 1
460  FOR Y = 1 TO 40:S(XC,Y) = 2: NEXT
470 A2 =  INT ( RND (1) * 71): IF  RND (
1) < 0.5 THEN A2 = 180 - A2
480 YC =  INT ( RND (1) * 60) + 1:B = YC
490  FOR X = 1 TO 40:L =  INT ( FN A(X))
500  IF X = XC THEN 520
510  FOR Y = 1 TO 40:T =  SGN (L - Y):S(
X,Y) = T + 2: NEXT
520  NEXT : GOTO 250
530 I =  PEEK ( - 16384):I = I - 200
540  PRINT I: GOTO 530
```

TREASURE NIM

To win this game, written by Dilwyn Jones, you must remove the very last chest. You have the option to remove one, two or three chests at each turn. The computer usually wins, but with a little study of the program you might win one or two games from time to time.

You are told throughout the game how many chests are left, and how many the computer took on its last turn.




```
10  REM  *** TREASURE NIM
20  REM  *** D.JONES
30  REM  *** MODIFIED BY B.ENGELHARDT
40  HOME
50 P =  INT ( RND (1) * 25) + 10
60  VTAB 1: HTAB 1: PRINT "*"
70  FOR C = 1 TO P - 2: VTAB 1: HTAB H +
 3
80  PRINT "+":H = H + 1: NEXT
90 H = H + 3
100  VTAB 5: HTAB 5: PRINT P;" CHESTS"
110  IF  RND (1) < .5 THEN 210
120  REM
130  VTAB 12: PRINT "HOW MANY CHESTS WIL
L YOU REMOVE";
140  GET R$:R =  VAL (R$)
150  IF R < 1 OR R > 3 THEN 140
160 P = P - R
170  VTAB 5: HTAB 5: PRINT P;" CHEST";:
IF P <  > 1 THEN  PRINT "S";
180  PRINT " LEFT   "
190  GOSUB 320
200  IF P < 1 THEN 310
210 N =  INT (P / 4) * 4
220  IF P <  > N THEN R = P - N
230 P = P - R
240  VTAB 9: PRINT "I REMOVED ";R
250  GOSUB 320
260  VTAB 5: HTAB 5: PRINT P;" CHEST";:
IF P <  > 1 THEN  PRINT "S";
270  PRINT " LEFT    "
280  IF P < 1 THEN 300
290  GOTO 120
300  VTAB 1: PRINT "$$$ I WIN $$$": FOR
TI = 1 TO 1900: NEXT : RUN
310  VTAB 1: PRINT "$$$ YOU WIN $$$": FO
R TI = 1 TO 1900: NEXT : RUN
320  FOR DE = 1 TO R: VTAB 1: HTAB H: PR
INT " "
330 H = H - 1: NEXT : RETURN
```

DUEL

In this exciting space battle, your ship and a robot ship are trapped inside an invisible force-field. Your ship occupies the left side of the screen and the robot ship occupies the right.

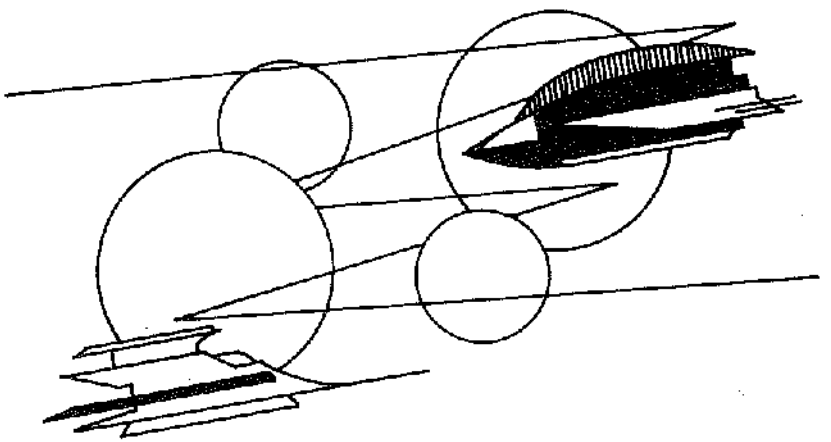
As you patrol your side of the force-field, you can use your laser cannon (that is, the space bar) to try and blast him out of the universe. Meanwhile, he relentlessly tracks you and homes in on you with his laser. Both your ships can take 15 direct hits with a laser cannon — after that, it's all over.

To control your ship, use the A key to move up, Z to move down and the space bar to fire.

A couple of useful hints to remember while playing this game are:

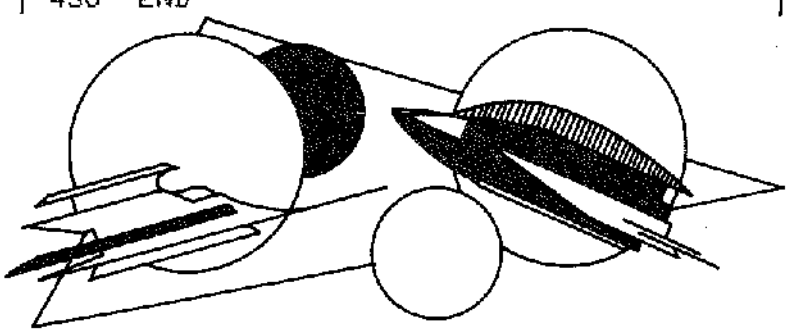
1) If you fire and miss while you are within a couple of spaces of the enemy ship, you are likely to get hit.

2) If you try to cross the enemy's path, you are likely to get hit UNLESS you hit him first.



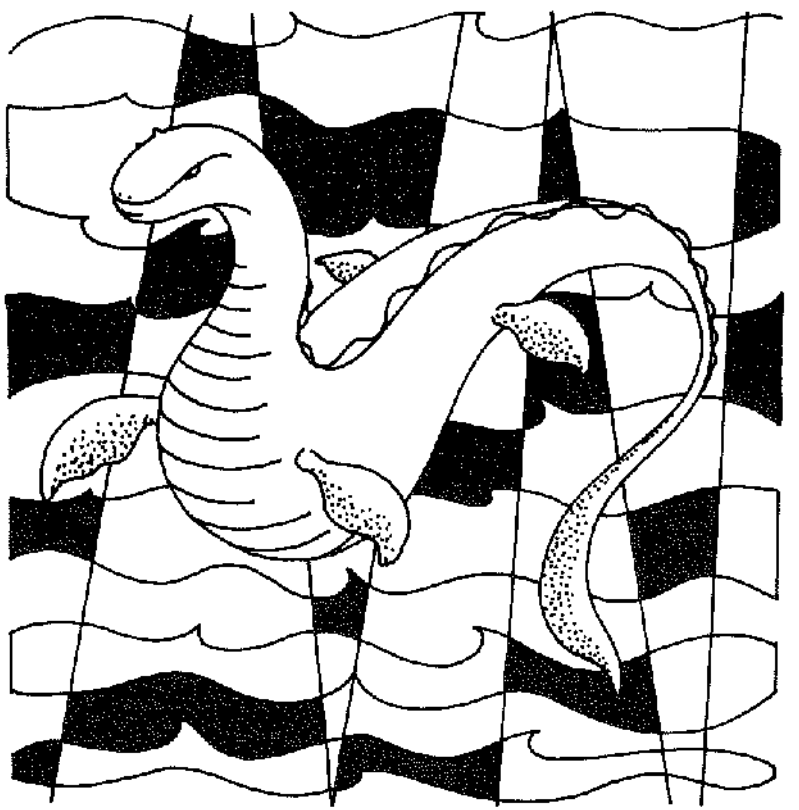

```
10  REM  DUEL
20  REM  TONY DYSON '84
30  HOME
40 S = 0:T = 0: POKE 34,1: HOME : GOTO 3
20
50  FOR I = 1 TO 37:F$ = F$ + "-": NEXT
60  GOSUB 310:P = V: GOSUB 310:Q = V
70  VTAB P + 2: HTAB 1: PRINT ">"
80  VTAB Q + 2: HTAB 39: PRINT "<"
90  GOSUB 390
100 KI =  PEEK ( - 16384): IF KI = 193 O
R KI = 218 THEN M = KI
110  IF M = 0 THEN 100
120  IF KI <  > 160 THEN 190
130  VTAB P + 2: HTAB 2: PRINT F$;: CALL
 768: IF P = Q THEN  PRINT "*"
140  IF P = Q THEN  GOSUB 380:S = S + 1:
 GOSUB 310
150  IF P = Q THEN Q = V: GOSUB 390:PU =
 1
160  VTAB P + 2: HTAB 1: CALL  - 868: PR
INT ">": VTAB Q + 2: HTAB 39: PRINT
     "<"
170 CK =  PEEK ( - 16368)
180  IF PU = 1 THEN PU = 0:M = 0: GOTO 1
00
190  IF M = 193 AND P > 0 THEN  VTAB P +
 2: HTAB 1: PRINT " ":P = P - 1
200  IF M = 193 AND P > 0 THEN  VTAB P +
 2: PRINT ">": GOTO 230
210  IF M = 218 AND P < 21 THEN  VTAB P
+ 2: HTAB 1: PRINT " ":P = P + 1
220  IF M = 218 AND P < 21 THEN  VTAB P
+ 2: PRINT ">"
230  IF Q <  > P THEN 270
240  CALL 768: VTAB Q + 2: PRINT "*";F$:
 GOSUB 380:T = T + 1: VTAB Q + 2
250  HTAB 1: CALL  - 868: HTAB 39: PRINT
 "<": GOSUB 310:P = V
```

```
260  VTAB P + 2: PRINT ">": GOSUB 390:M
= 0: GOTO 100
270  IF Q < P THEN Q = Q + 1: VTAB Q + 1
: HTAB 39: PRINT " ": VTAB Q + 2
280  IF Q < P THEN  HTAB 39: PRINT "<":
GOTO 100
290 Q = Q - 1: VTAB Q + 3: HTAB 39: PRIN
T " ": VTAB Q + 2: HTAB 39
300  PRINT "<": GOTO 100
310 V =  INT ( RND (1) * 23): RETURN
320 A1$ = "300:A9 10 85 10 A2 D0 C6 10 D
0 07 A9 10 85 10 4C 14 "
330 A2$ = "03 AD 30 C0 20 1B 03 CA D0 EC
 60 A4 11 88 D0 FD "
340 A3$ = "60 ND823G"
350 A$ = A1$ + A2$ + A3$
360  FOR I = 1 TO  LEN (A$): POKE 511 +
I, ASC ( MID$ (A$,I,1)) + 128
370  NEXT : POKE 72,0: CALL  - 144: GOTO
 50
380  FOR I = 1 TO 200:SN =  PEEK ( - 163
36): NEXT : RETURN
390  POKE 34,0: VTAB 1: PRINT  TAB( 10);
S; TAB( 30);T
400  POKE 34,1: IF S = 15 OR T = 15 THEN
 420
410  RETURN
420  INPUT "ANOTHER GAME ? ";YN$: IF YN$
 = "Y" THEN  RUN
430  END
```



NESSIE

The Loch Ness monster is hiding behind one of nine rocks on the shore of the Loch. To locate the monster you press the key corresponding to the rock. If a '*' appears you must press any key as quickly as possible or else Nessie has another feed!

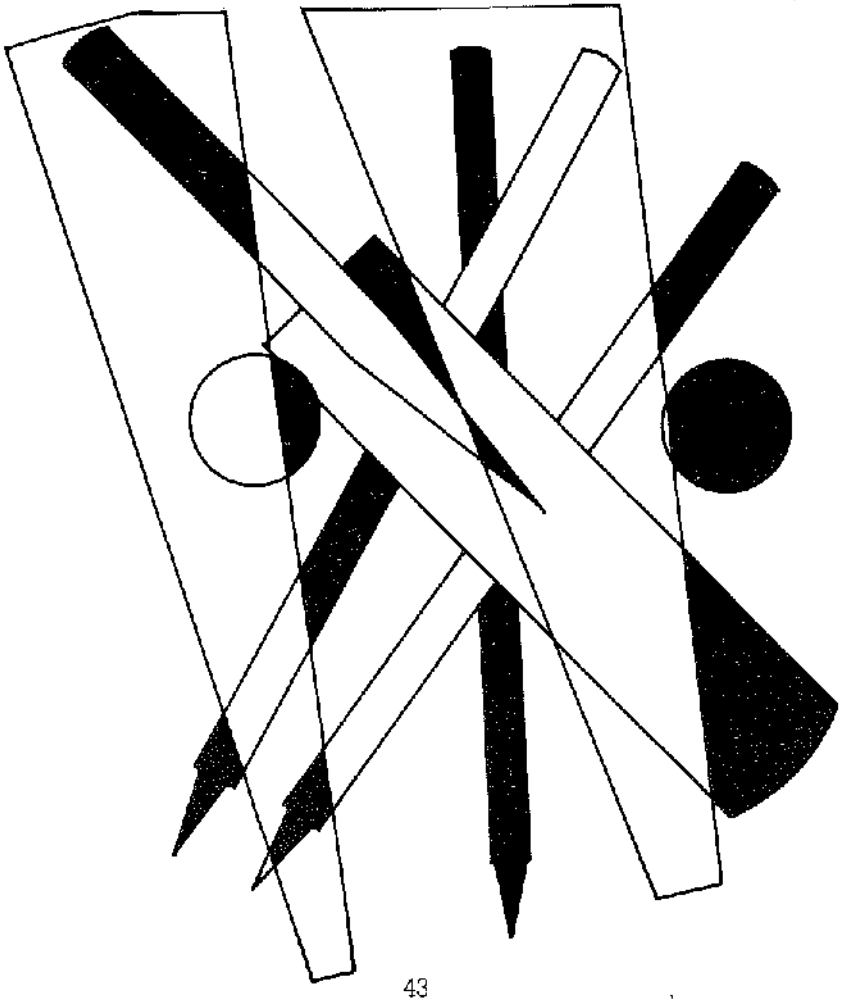


```
10  REM  *** NESSIE
20  REM  *** D.JONES
30  REM  *** ADAPTED BY B.ENGELHARDT
```

```
40  HOME
50 U$ =  STR$ ( INT ( RND (1) * 9) + 1)
60  HTAB 10:H = 10
70  VTAB 3: INVERSE : FOR L = 1 TO 9: PR
INT L;:H = H + 2: HTAB H
80  NEXT : NORMAL
90  VTAB 6: HTAB 17: PRINT "!-!": HTAB 1
7: PRINT "YOU"
100  GET A$
110  IF A$ < "1" OR A$ > "9" THEN 100
120  VTAB 3: HTAB 8 + ( VAL (A$) * 2): P
RINT " "
130  POKE  - 16384,0
140  FOR I = 1 TO  INT ( RND (1) * 30) +
 30
150  IF  PEEK ( - 16384) > 127 THEN 300
160  NEXT I
170  IF A$ <  > U$ THEN 100
180  VTAB 3: HTAB 8 + ( VAL (U$) * 2): P
RINT "*"
190  POKE  - 16384,0
200  FOR I = 1 TO 35: IF  PEEK ( - 16384
) > 127 THEN 280
210  NEXT I
220  VTAB 20: HTAB 14: INVERSE : PRINT "
YUM  YUM";: NORMAL : HTAB 25
230  PRINT "WHO'S A BIT SLOW"
240  VTAB 12: PRINT "ANOTHER GO";: GET A
$: GET A$
250  IF A$ <  > "Y" THEN  END
260  HOME
270  RUN
280  VTAB 3: HTAB 8 + ( VAL (U$) * 3 - 2
): PRINT "GOT IT"
290  GOTO 240
300  VTAB 1: HTAB 10: PRINT "TRIGGER HAP
PY PERSON"
310  GOTO 240
```


HOWZAT

The stage is set for a showdown between you and your opposition. You play as many overs as you like, but once you are out your score returns to zero. The computer umpires the game and also prints a scoreboard after each over. After the game the winner will be displayed.



```
10  REM  *** HOWZAT
20  REM  *** B.ENGELHARDT
30  HOME : VTAB 12: HTAB 13: PRINT "--HO
WZAT--"
40  FOR A = 1 TO 1500: NEXT : GOSUB 940
50  HOME : VTAB 5
60  INPUT "PLAYER ONE'S NAME:";P1$
70  VTAB 7: INPUT "PLAYER TWO'S NAME:";P
2$
80  GOSUB 940
90  PRINT
100 TS =  INT (10 *  RND (1)) + 1
110  IF TS > 5 THEN T$ = "R": GOTO 130
120 T$ = "F"
130  PRINT P1$;"'S CALL <F>LATS  <R>OOF-
TOPS";: GET S$
140  PRINT
150  IF S$ = T$ THEN  PRINT P1$;" WON TH
E TOSS"
160  IF S$ = T$ THEN  INPUT "WHAT WILL Y
OU DO BAT/BOWL ";CH$
170  IF S$ = T$ AND CH$ = "BOWL" THEN O1
 = 0:02 = 1:03 = 1:04 = 0
180  IF S$ = T$ AND CH$ = "BAT" THEN O1
= 1:02 = 0:03 = 0:04 = 1
190  IF S$ = T$ THEN 230
200  PRINT P2$;" WON THE TOSS": INPUT "W
HAT WILL YOU DO BAT/BOWL ";CH$
210  IF CH$ = "BAT" THEN O1 = 0:02 = 1:0
3 = 1:04 = 0: GOTO 230
220 O1 = 1:02 = 0:03 = 0:04 = 1
230  GOSUB 940: INPUT "HOW MANY OVERS:";
OV
240 O = O + 1: IF O > OV THEN 520
250 BA = BA + 1: IF BA > 6 THEN 640
260  PRINT : PRINT "WHAT TYPE OF BOWL ";
270  IF O4 = 1 AND O1 = 1 THEN 300
280  PRINT P1$
290  GOTO 310
300  PRINT P2$
```

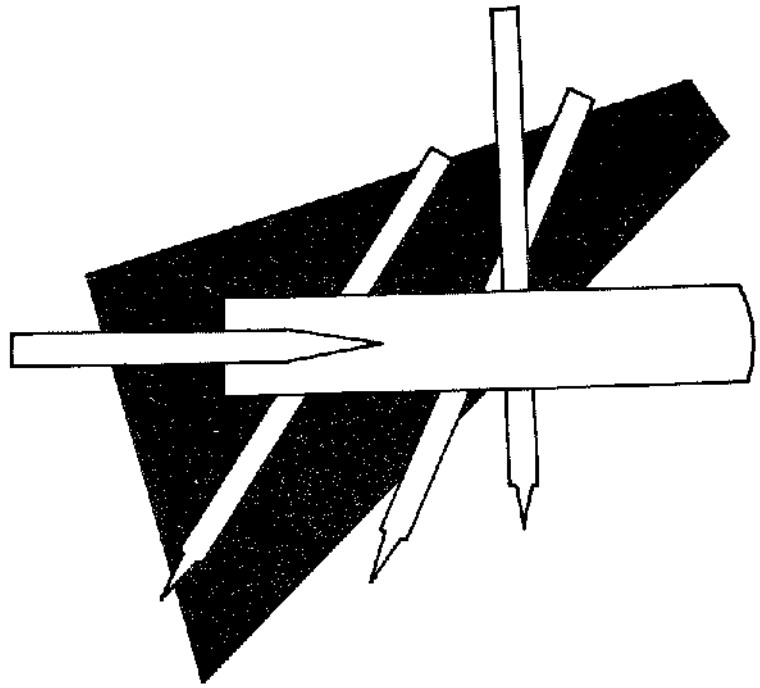


```
310  PRINT "(1)FAST  (2)MEDIUM  (3)SLOW
 (4)RANDOM"
320  GOSUB 760
330  GET BO: IF BO > 4 THEN 330
340  ON BO GOTO 360,380,390,350
350 RN =  INT (3 *  RND (1)) + 1: ON RN
GOTO 360,380,390
360  REM
370  PRINT " BOWLS A QUICKIE ":RN = 12:
GOTO 400
380  PRINT "SENDS DOWN A MEDIUM PACER ":
RN = 20: GOTO 400
390  PRINT " TRIES A SLOW BOWL ":RN = 30
400  PRINT
410  GOSUB 870
420  IF RN - SP >  INT (RN *  RND (1)) +
 1 THEN 470
430  INVERSE : FLASH : PRINT "IS OUT FOR
 ";: NORMAL
440  IF O1 = 1 AND O4 = 1 THEN  PRINT P1
;" RUNS": GOTO 610
450  PRINT P2;" RUNS"
460  GOTO 610
470  PRINT " GETS ";:RU =  INT (4 *  RND
 (1)) + 1: PRINT RU
480  IF O1 = 1 AND O4 = 1 THEN P1 = P1 +
 RU: GOTO 500
490 P2 = P2 + RU
500  GOTO 250
510  GOTO 240
520  GOSUB 940
530 RO = RO + 1
540 O = 0
550  PRINT "END OF INNINGS !!!!"
560  IF RO > 1 THEN 960
570  PRINT : PRINT "GET READY TO START"
580  FOR TT = 1 TO 2000: NEXT
590  IF O1 = 1 AND O4 = 1 THEN O1 = 0:02
 = 1:03 = 1:04 = 0: GOTO 240
```

```
600 01 = 1:02 = 0:03 = 0:04 = 1
610  REM
620  IF 04 = 1 AND 01 = 1 THEN P1 = 0: G
OTO 250
630 P2 = 0: GOTO 250
640  HOME : VTAB 1: HTAB 15: PRINT "SCOR
E CARD"
650  VTAB 3
660  PRINT P1$: VTAB 3: HTAB 15: PRINT P
1: VTAB 3: HTAB 20: PRINT P2$: VTAB
     3: HTAB 35: PRINT P2
670  PRINT : PRINT : PRINT
680 BA = 0
690  IF P1 > P2 THEN  PRINT P1$;" LEADS
BY ";(P1 - P2);: GOTO 710
700  PRINT P2$;" LEADS BY ";(P2 - P1);
710  PRINT " RUNS"
720  PRINT : PRINT "OVERS:";0
730 T1 = 0:T2 = 0
740  VTAB 10: PRINT "PRESS ANY KEY TO CO
NTINUE-";: GET A$: PRINT :
750  GOTO 510
760  IF 01 = 1 AND 04 = 1 THEN 780
770  PRINT P1$;: RETURN
780  PRINT P2$;: RETURN
790  END
800  IF 04 = 1 AND 01 = 1 THEN 830
810  PRINT P1$;
820  RETURN
830  PRINT P2$;: RETURN
840  IF 02 = 1 AND 03 = 1 THEN 860
850  PRINT P1$;: RETURN
860  PRINT P2$;: RETURN
870  PRINT
880  PRINT "WHAT TYPE OF SHOT ";: GOSUB
840: PRINT : PRINT "(1)ATTACKING
      (2)CAUTIOUS  (3)DEFENSIVE"
890  GET SH$:SH =  VAL (SH$)
900  ON SH GOTO 910,920,930
910  GOSUB 840: PRINT " PLAYS AN ATTACKI
```



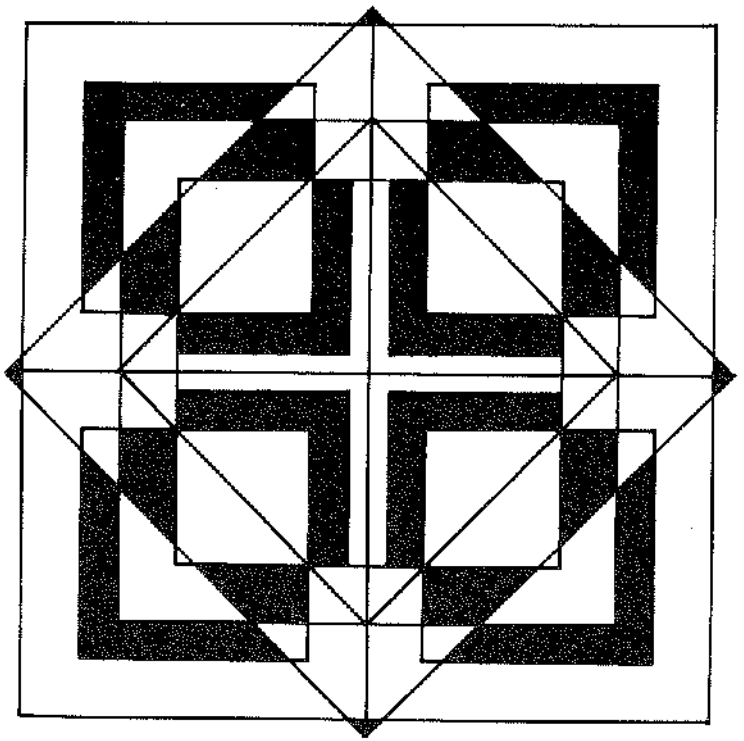
```
NG SHOT AND ";: RETURN
920  GOSUB 840: PRINT " PLAYS VERY CAUTI
OUSLY AND ";: RETURN
930  GOSUB 840: PRINT " IS VERY DEFENSIV
E AND ";: RETURN
940  FOR II = 1 TO 20: POKE 32,20 - II:
POKE 33,2 * II: HOME
950  FOR TT = 1 TO 25: NEXT : NEXT : RET
URN
960  GOSUB 940
970  VTAB 1: PRINT "GAME OVER !!!!!!!":
PRINT
980  IF P1 > P2 THEN  PRINT P1$;" WON BY
 ";(P1 - P2);" RUNS": END
990  IF P1 = P2 THEN  PRINT "A TIE !!!!!
": END
1000  PRINT P2$;" WON BY ";(P2 - P1);" R
UNS": END
```



SIMON SAYS

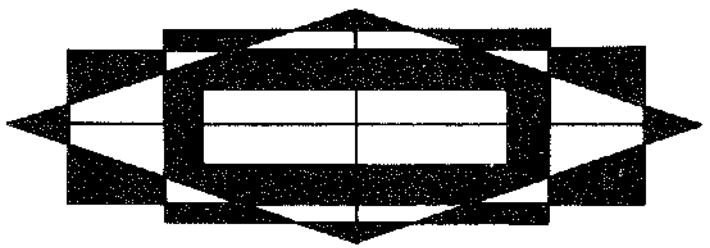
How good is your memory? Can you remember a sequence of six, seven or maybe even eight colours? The game starts with one colour and another one is added after each correct turn. The controls are as follows:

Top = 1
Right = 2
Bottom = 3
Left = 4




```
10  REM  *** SIMON SAYS
20  REM  *** T.HARTNELL
30  REM  *** MODIFIED BY B.ENGELHARDT
40  GOSUB 540
50  GR
60  COLOR= 15
70  HLIN 16,25 AT 9: HLIN 11,30 AT 15: H
LIN 11,30 AT 21
80  HLIN 16,25 AT 27
90  VLIN 9,27 AT 16: VLIN 9,27 AT 25: VL
IN 15,21 AT 11
100  VLIN 15,21 AT 20: VLIN 15,21 AT 21:
 VLIN 15,21 AT 30
110  DIM SQ(20)
120 LEV = LEV + 1
130  IF LEV > 20 THEN  END
140  HOME : INVERSE : VTAB 22: HTAB 18:
PRINT "LOOK"
150  FOR TI = 1 TO 1500: NEXT
160  FOR CD = 1 TO LEV
170 SQ(CD) =  INT ( RND (1) * 4) + 1
180  IF SQ(CD) = S1 THEN 170
190 S1 = SQ(CD)
200 CO = SQ(CD): GOSUB 350
210  NEXT
220  VTAB 22: HTAB 15: PRINT "YOUR TURN"
230  FOR TI = 1 TO 1500
240  NEXT
250  FOR US = 1 TO LEV
260  GET K$:K =  VAL (K$): IF K < 1 OR K
 > 4 THEN 260
270 CO = K: GOSUB 350
280  IF K <  > SQ(US) THEN 330
290  NEXT
300  HTAB 15: VTAB 22: PRINT "VERY GOOD"
310  FOR TI = 1 TO 1500: NEXT
320  GOTO 500
330  HOME : VTAB 22: HTAB 17: PRINT "WRO
NG": FOR TI = 1 TO 1400: NEXT
340  GOTO 140
```

```
350  IF CO = 1 THEN A = 17:B = 24:C = 10
:D = 14:E = 2
360  IF CO = 2 THEN A = 22:B = 29:C = 16
:D = 20:E = 13
370  IF CO = 3 THEN A = 17:B = 24:C = 22
:D = 26:E = 4
380  IF CO = 4 THEN A = 12:B = 19:C = 16
:D = 20:E = 11
390  COLOR= E
400  FOR PL = A TO B
410  VLIN C,D AT PL
420  NEXT
430  POKE 0,30 + (CO * 10): POKE 1,70: C
ALL 771
440  FOR TI = 1 TO 400: NEXT
450  COLOR= 0
460  FOR DE = A TO B
470  VLIN C,D AT DE
480  NEXT
490  RETURN
500  COLOR= 8
510  PLOT L1,35
520 L1 = L1 + 2
530  GOTO 120
540  FOR PO = 771 TO 789: READ Z: POKE P
O,Z: NEXT : RETURN
550  DATA  173,48,192,136,208,4,198,1,24
0,8,202,208,246
560  DATA 166,0,76,3,3,96
```



GOLF

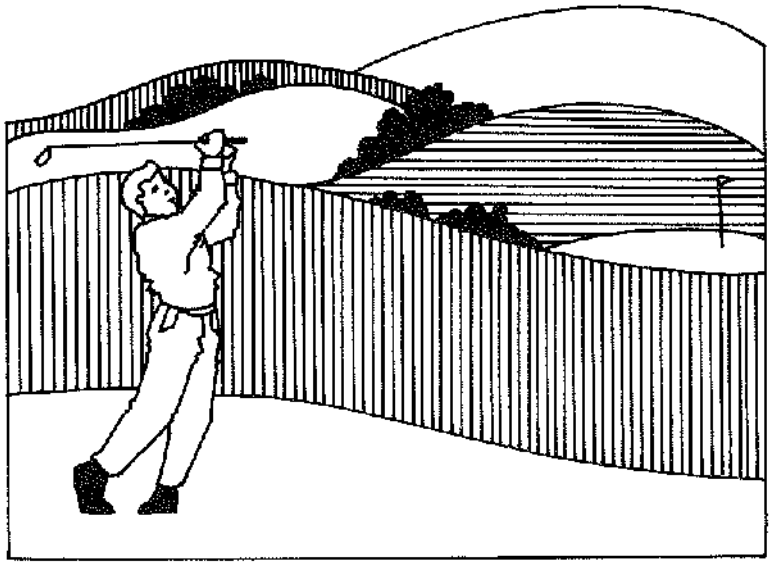
You are entered in a tournament and you can play as many holes as you desire. You have two sets of clubs. The first is for use on the fairway. The second is for the area around and on the green. The computer will tell you your par for each hole and the distance to the hole. At the end you will be told your par for the holes played, and the total par.

```
10  REM  *** GOLF
20  REM  *** A.GOURLAY
30  REM  *** ADAPTED BY B.ENGELHARDT
40  HOME
50 N =  RND ( - TI)
60  DEF  FN R(X) =  INT ( RND (1) * X)
70  GOSUB 530
80 TP = 0:TS = 0:HO = 1
90  HOME : PRINT "YOU ARE ON TEE ";HO:S
= 0
100 D =  FN R(250) + 100:P =  INT (D / 6
0): IF P < 2 THEN P = 2
110 TP = TP + P
120  PRINT "HOLE IS ";D;" METERS LONG":
PRINT "PAR ";P
130  IF S = 0 THEN 160
140  PRINT "DISTANCE TO HOLE ";D;" METER
S": PRINT "SHOTS ";S
150  IF D > 40 THEN  PRINT "IN THE FAIRW
AY"
160  PRINT
170  IF D < 20 THEN  PRINT "ON THE GREEN
": GOTO 190
180  IF D < 40 THEN  PRINT "IN SIGHT OF
THE GREEN"
190  INVERSE : FOR Z = 1 TO 26: PRINT "
";: NEXT : NORMAL
```

```
200  PRINT : PRINT
210  PRINT "SELECT CLUB STRENGTH (1-9)";
220  GET D$: IF D$ < "1" OR D$ > "9" THE
N 220
230  PRINT
240 S = S + 1
250 D1 =  VAL (D$) * (25 +  FN R(15)): I
F D < 40 THEN D1 =  INT (D1 / 6)
260  IF D < 5 AND D$ < "3" THEN D1 = D
270  IF D1 = 0 THEN D1 = 1
280  PRINT "YOU HIT ";D1;" METERS"
290 D =  ABS (D - D1): IF D <  = 1 THEN
320
300  FOR N = 1 TO 2000: NEXT
310  GOTO 130
320  REM  ***
330  REM
340  IF S > 1 THEN 370
350  PRINT "OOW !!!!"
360  GOTO 400
370  INVERSE : FOR Z = 1 TO 26: PRINT "
";: NEXT : NORMAL
380  PRINT : PRINT
390  PRINT "YOU TOOK ";S;" SHOTS"
400  PRINT "THEN PAR WAS ";P
410 TS = TS + S
420  PRINT "SHOTS SO FAR ";TS: PRINT : P
RINT "TOTAL PAR ";TP
430  PRINT
440  REM
450 HO = HO + 1: IF HO > H1 THEN 470
460  GOSUB 620: GOTO 90
470 TP = TS - TP: IF TP = 0 THEN  PRINT
"YOU EQUALLED PAR": GOTO 510
480  PRINT "YOU COMPLETED THE COURSE ";
ABS (TP);
490  IF TP > 0 THEN  PRINT " OVER"
500  IF TP < 0 THEN  PRINT " UNDER"
510  PRINT : GOSUB 620
520  RUN
```



```
530  REM  *** INSTRUCTIONS
540  PRINT "WHEN YOU ARE ON THE FAIRWAY
YOUR CLUBS "
550  PRINT : PRINT "ARE STRONG ENOUGH TO
 HIT AROUND 300 M"
560  PRINT
570  PRINT "WHEN YOU ARE ON OR IN SIGHT
OF THE GREEN"
580  PRINT "YOUR CLUBS ONLY HIT AROUND 6
0 M"
590  PRINT
600  INPUT "HOW MANY HOLES ";H1
610  PRINT
620  HTAB 13: PRINT "PRESS ANY KEY";
630  GET D$
640  PRINT
650  PRINT : PRINT : PRINT
660  RETURN
```



PHASAR

An invading alien — the rectangular object on the right — is threatening to overtake your planet. Your duty is to defend the planet by firing your phasar at the alien. You do this by pressing the space bar.

You have 25 shots in which to get the highest possible hit/miss ratio. You also have 15 speeds to choose from.

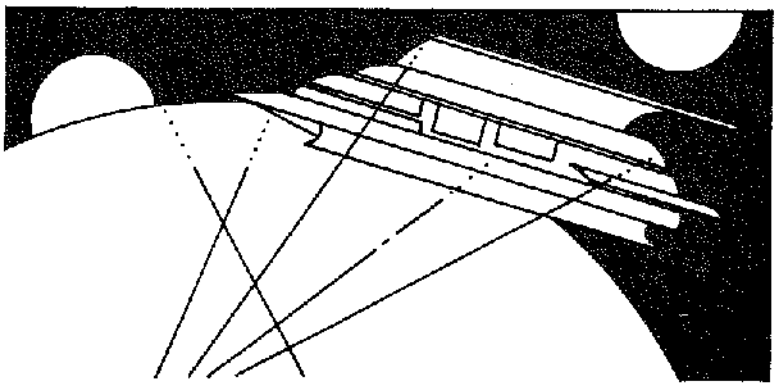
Before playing the game you might need to turn the computer off as the program uses a lot of machine code for its sounds and shapes.

```
10  REM  *** PHASAR
20  REM  *** B.ENGELHARDT
30  GOSUB 400
40  GOSUB 620
50  GOSUB 740
60 Y1 = 12
70  HGR
80  HCOLOR= 3: HPLOT 0,0 TO 279,0 TO 279
,157 TO 0,157 TO 0,0
90  HOME
100  VTAB 24
110  INPUT "WHAT SPEED 1-15 ";S
120  IF S < 1 OR S > 15 THEN 90
130  HOME
140  SCALE= 1
150 Y = 70
160  HCOLOR= 3
170  DRAW 1 AT 10,Y
180  GOSUB 520
190 K =  PEEK ( - 16384)
200  IF K = 160 THEN 260
210  XDRAW 1 AT 10,Y
220  IF Y <  = 15 THEN Y = 150
230 Y = Y - S
240  GOSUB 560
```



```
250  GOTO 170
260  HPLOT 13,Y - 4 TO 279,Y - 4
270 SH = SH + 1
280  IF SH >  = 25 THEN 700
290  FOR SO = 18 TO 12 STEP  - 1
300  POKE 0,SO: POKE 1,SO
310  CALL 771
320  NEXT
330  HCOLOR= 0
340  IF Y >  = Y1 AND Y1 >  = (Y - 15) T
HEN  GOSUB 670
350  HPLOT 13,Y - 4 TO 278,Y - 4
360  HCOLOR= 3
370  POKE  - 16368,0
380  GOSUB 480
390  GOTO 240
400  POKE 233,96: POKE 232,0
410  FOR A = 24576 TO 24614
420  READ D
430  POKE A,D
440  NEXT A
450  RETURN
460  DATA  1,0,4,0,63,63,63,63,36,36,45,
45,45,0,1,0,4,0,45,36
470  DATA  45,54,45,54,54,54,62,55,62,39
,60,39,36,36,36,0,0,0,0
480  IF SH = 0 THEN  RETURN
490 PH =  INT ((HI / SH) * 100)
500  HOME : VTAB 24: PRINT "HIT/MISS %:"
;PH;"%"
510  RETURN
520  POKE 232,14: POKE 233,96
530  DRAW 1 AT 270,Y1
540  POKE 232,0: POKE 233,96
550  RETURN
560  POKE 232,14: POKE 233,96
570  XDRAW 1 AT 270,Y1
580 Y1 = Y1 + (5 + S)
590  IF Y1 >  = 149 THEN Y1 = 13
600  POKE 232,0
```

```
610  RETURN
620  FOR Z = 790 TO 829: READ X: POKE Z,
X: NEXT : DATA  169,208,133
630  DATA  29,69,30,133,28,234,234,101,2
8,112,3,174,48,192,200,208
640  DATA  238,230,29,174,48,192,230,27,
165,27,201,64,144,225
650  DATA   230,30 ,96,0,0,0,0
660  RETURN
670  FOR AE = 1 TO 50: CALL 795: NEXT
680 HI = HI + 1
690  RETURN
700  HOME : VTAB 24: PRINT "YOUR HIT MIS
S RATIO WAS ";PH;"%"
710  INPUT "DO YOU WANT TO PLAY AGAIN ";
A$
720  IF A$ = "N" THEN  TEXT : END
730  RUN
740  FOR SP = 771 TO 789: READ Z: POKE S
P,Z: NEXT : RETURN
750  DATA  173,48,192,136,208,4,198,1,24
0,8,202,208,246
760  DATA  166,0,76,3,3,96
```

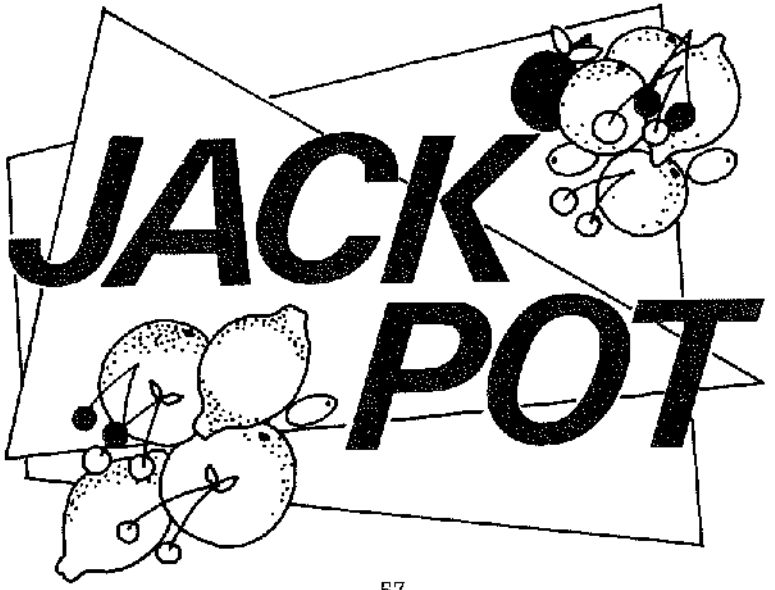


JACKPOT

This is one of the first games we wrote for the Apple. It uses a lot of POKEing rather than PRINT statements, since PRINT statements take nearly double the time. You start with $10 (which can easily be changed) and your object is to win as much as possible. The computer automatically deducts 50¢ for each game. The winning combinations, in order, are:

BBB (JACKPOT)
*** (ANY THREE THE SAME)
BB–
XX– (FIRST TWO THE SAME)
XXA

* = Same symbol
– = Any symbol

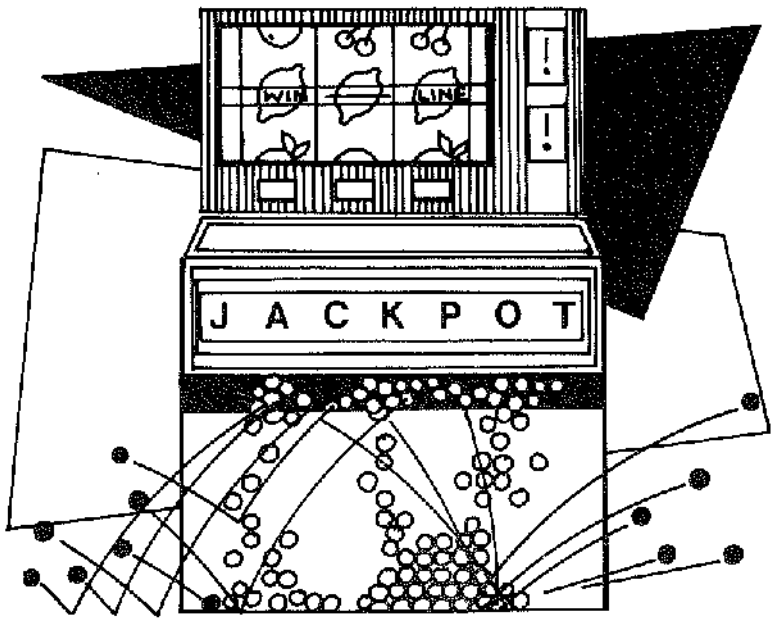


```
10  REM  *** JACKPOT
20  REM  *** B.ENGELHARDT
30 M = 10: REM  *** MONEY
40  GOSUB 700
50  HOME
60  FOR P = 1 TO 7: FOR L = 7 TO 13: POK
E 1800 + L, INT (200 *  RND (1)) +
     55: NEXT : POKE 0,10: POKE 1,5: CAL
L 771: NEXT
70  HOME : VTAB 7: HTAB 16: PRINT "JACKP
OT"
80  FOR D = 1 TO 1000: NEXT
90  INVERSE
100  VTAB 6: HTAB 15: PRINT "          "
110  VTAB 7: HTAB 15: PRINT " ": VTAB 7:
 HTAB 23: PRINT " "
120  VTAB 8: HTAB 15: PRINT "          "
130  FOR A = 9 TO 14: VTAB A: HTAB 15: P
RINT " ": VTAB A
140  HTAB 23: PRINT " "
150  NEXT
160  VTAB 15: HTAB 15: PRINT "          "
170  VTAB 13: HTAB 17: PRINT "+++++"
180  VTAB 10: HTAB 16: PRINT "* * * *"
190  GOSUB 600
200  NORMAL
210  VTAB 18: HTAB 15: PRINT "TOTAL $";M
 - .5;"    "
220  IF M = 0 THEN 630
230  VTAB 22
240  PRINT "PRESS ANY KEY TO PULL ARM-";
: CALL  - 756:M = M - .5
250  GOSUB 620: REM  *** ARM PULL
260  FOR D = 1 TO 500: NEXT
270  GOSUB 600
280  FOR A = 1 TO 20
290 W1 =  INT (9 *  RND (1)) + 193
300 W2 =  INT (9 *  RND (1)) + 193
310 W3 =  INT (9 *  RND (1)) + 193
320  POKE 0,A + A: POKE 1,A + A
```



```
330  POKE 1208,W1: POKE 1210,W2: POKE 12
12,W3
340  CALL 771
350  NEXT
360  IF W1 = 194 AND W2 = 194 AND W3 = 1
94 THEN W = 1:WI = 25: GOTO 450
370  IF W1 = W2 AND W1 = W3 THEN W = 1:W
I = 15: GOTO 450
380  IF W1 = 194 AND W2 = 194 THEN WI =
5: GOTO 450
390  IF W1 = W2 THEN WI = 2.5: GOTO 450
400  IF W3 = 193 THEN WI = 1: GOTO 450
410  VTAB 1: HTAB 15: PRINT "YOU  LOSE !
"
420  FOR DE = 1 TO 1500: NEXT
430  VTAB 1: HTAB 15: PRINT "
  "
440  GOTO 590
450 M = M + WI
460  IF W = 1 THEN  INVERSE : FLASH : VT
AB 7: HTAB 16: PRINT "JACKPOT"
470  VTAB 1: HTAB 15: PRINT "YOU WIN !"
480  NORMAL
490 W = 0
500  FOR SO = 1 TO (WI * 2)
510  POKE 0,40
520  POKE 1,25
530  FOR DE = 1 TO 40: NEXT
540  CALL 771
550  NEXT
560  FOR DE = 1 TO 2000: NEXT
570  VTAB 1: HTAB 15: PRINT "
    "
580  VTAB 7: HTAB 16: PRINT "JACKPOT"
590  GOTO 200
600  POKE 1088,163: POKE 1216,163: POKE
1344,163: POKE 1472,189
610  POKE 1471,189: POKE 1816,160: POKE
1944,0: RETURN
620  POKE 1088,160: POKE 1216,160: POKE
```

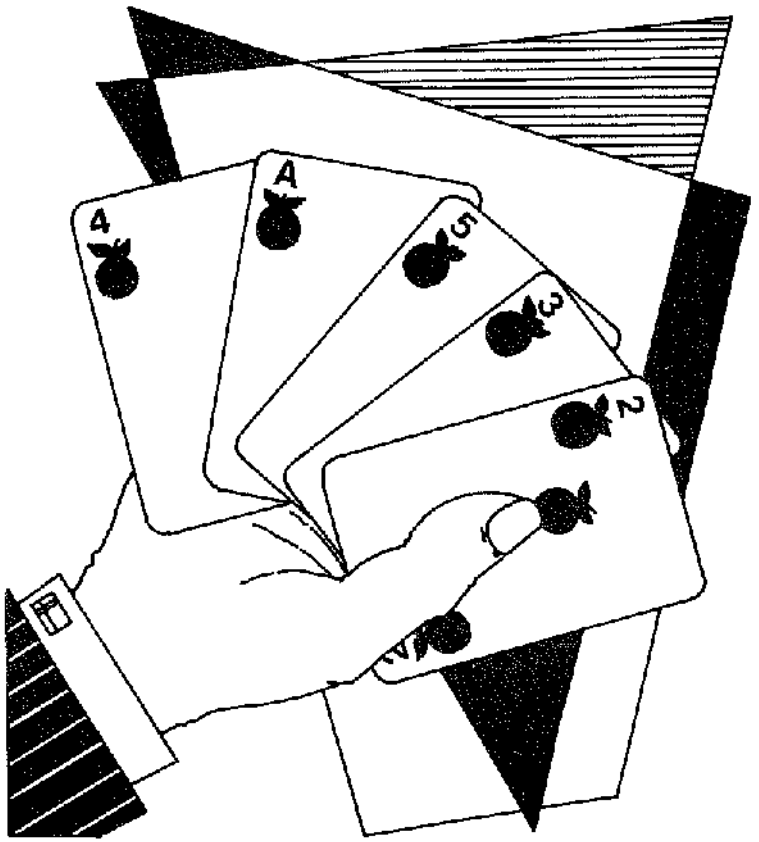
```
1816,160: POKE 1944,160: RETURN
630  HOME
640  VTAB 5: PRINT "YOU HAVE USED ALL YO
UR CHANGE. PRESS"
650  VTAB 7: PRINT "<0> TO LEAVE OR ANY
OTHER KEY TO CHANGE"
660  VTAB 9: PRINT "ANOTHER $10";: GET K
$
670  IF K$ = "0" THEN  END
680 M = 10
690  GOTO 70
700  FOR SP = 771 TO 789: READ Z: POKE S
P,Z: NEXT : RETURN
710  DATA  173,48,192,136,208,4,198,1,24
0,8,202,208,246
720  DATA  166,0,76,3,3,96
```



TWENTY ONE

In this classic game, the computer is the dealer, and you are the opponent. Taking it in turns, you both get the opportunity to 'draw a card' out of the 'pack'. The idea is to get your total as close to 21 as possible without exceeding it.

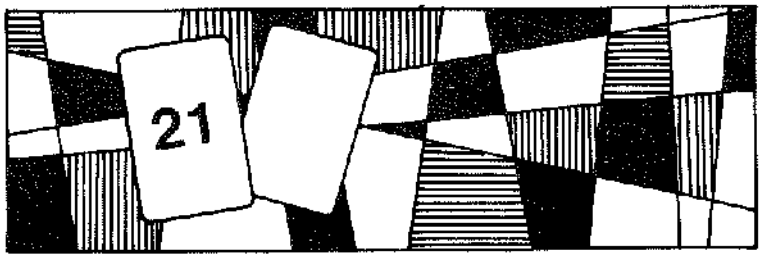
Aces count as 1, number cards get face value, and court cards get 10.



```
10  REM  TWENTY ONE
20  REM  TONY DYSON '84
30  DIM P$(12),C$(12),PA$(52):PC = 1:PT
= 0:CT = 0:CC = 0
40  FOR I = 1 TO 4: RESTORE
50  FOR J = 1 TO 13: READ N$: ON I GOTO
60,70,80,90
60 PA$(I * J) = N$ + " HEARTS": NEXT : G
OTO 100
70 PA$(I * J) = N$ + " DIAMONDS": NEXT :
 GOTO 100
80 PA$(I * J) = N$ + " CLUBS": NEXT : GO
TO 100
90 PA$(I * J) = N$ + " SPADES": NEXT
100  NEXT : HOME
110  IF  RND (1) > 0.57 THEN  PRINT "DEA
LER GOES FIRST.": PRINT : GOTO 20
     0
120  PRINT "YOU GO FIRST.": PRINT
130  INPUT "DO YOU WISH TO DRAW A CARD?
";YN$
140  IF YN$ = "Y" THEN P = 1: GOSUB 320:
PC = PC + 1:P = 0: GOTO 160
150 PA = 1
160  HOME : PRINT : PRINT : PRINT : PRIN
T "YOUR HAND:": PRINT
170  FOR O = 1 TO PC: PRINT  TAB( 7);P$(
O): NEXT
180  PRINT  TAB( 7);"---------": PRINT "
TOTAL:";PT
190  PRINT  TAB( 7);"---------": PRINT :
 PRINT : IF PT >  = 21 THEN 250
200  IF CT > 17 AND  RND (1) < 0.85 THEN
 240
210  PRINT "DEALER DRAWS A CARD.": PRINT
 :C = 1: GOSUB 320:CC = CC + 1
220  IF CT >  = 21 THEN 250
230 PA = 0: GOTO 130
240  PRINT "DEALER PASSES.": PRINT : IF
PA <  > 1 THEN 130
```



```
250  PRINT "YOU:";PT: PRINT "DEALER:";CT
260  IF ((PT > CT) AND (PT < 22)) OR CT
> 21 THEN 440
270  IF PT = CT THEN  PRINT "YOU DREW TH
E GAME!": GOTO 290
280  PRINT "YOU LOST."
290  PRINT : INPUT "ANOTHER HAND? ";YN$
300  IF YN$ = "Y" THEN  RUN
310  END
320 R =  INT ( RND (1) * 52) + 1
330  IF R = 53 THEN R = 1
340  IF PA$(R) = "" THEN R = R + 1: GOTO
 330
350  RESTORE : FOR I = 1 TO 13: READ A$
360  IF  LEFT$ (PA$(R),2) =  LEFT$ (A$,2
) THEN 380
370  NEXT
380  IF I > 10 THEN I = 10
390  IF C = 1 THEN 420
400 PT = PT + I:P$(PC) = PA$(R)
410 P = 0: GOTO 430
420 CT = CT + I:C$ = PA$(R):C = 0
430 PA$(R) = "": RETURN
440  PRINT : PRINT "YOU WON": GOTO 290
450  DATA  ACE,TWO,THREE,FOUR,FIVE,SIX,S
EVEN
460  DATA  EIGHT,NINE,TEN,JACK,QUEEN,KIN
G
```



SONIC FORCE

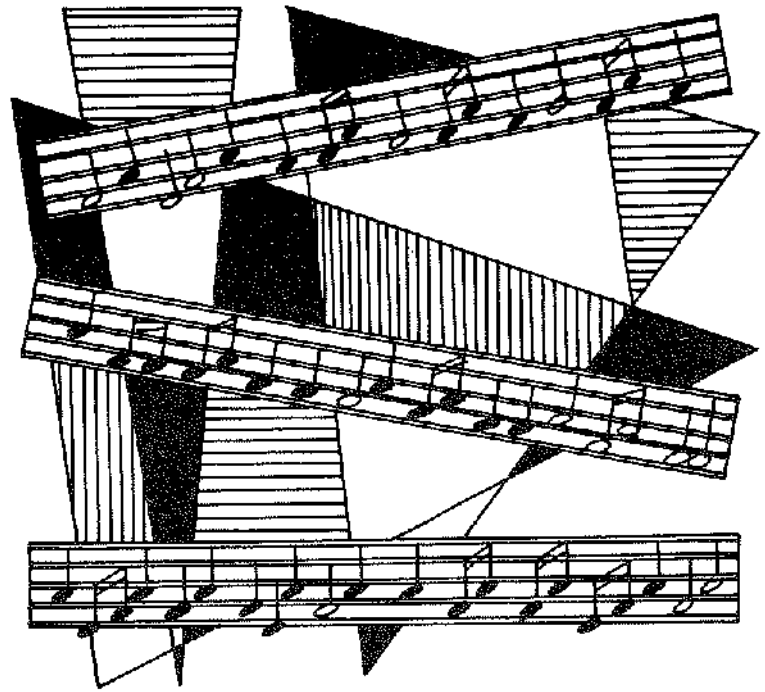
You'll have to concentrate if you want to succeed at this game of sharp hearing. The computer plays one of 255 random tones, and your job is to try and guess which one. The higher the number, the higher the pitch. To give you an idea of approximately which number corresponds to which note, some random notes are played just before you make your guess each time.

The idea for this game came from Clive Gifford's version for the Dragon. Good luck!

```
10  REM  SONIC FORCE
20  REM  TONY DYSON '84
30  REM  IDEA:CLIVE GIFFORD (FOR DRAGON)
40  REM  SONIC FORCE
50  GOSUB 290
60  TEXT : HOME :A$ = "**************"
70  HTAB 15: INVERSE : PRINT A$: HTAB 15
: PRINT "*SONIC FORCE*"
80  HTAB 15: PRINT A$: NORMAL
90  VTAB 10: PRINT "HERE ARE SOME RANDOM
 TONES..."
100  PRINT : PRINT : PRINT "<HIT ANY KEY
>": GET A$
110 I = 0
120 P =  INT ( RND (1) * 255) + 1: POKE
0,255 - P: POKE 1,100
130  PRINT P: CALL 771
140  FOR J = 1 TO 100: NEXT
150 I = I + 1: IF I < 6 THEN 120
160  PRINT : INPUT "MORE RANDOM TONES";Y
N$:
170  IF YN$ = "Y" THEN  PRINT : GOTO 110
180  PRINT : PRINT "HERE COMES THE MYSTE
RY TONE..."
190  FOR I = 1 TO 300: NEXT
```



```
200 P =  INT ( RND (1) * 255) + 1: POKE
0,255 - P: POKE 1,150
210  CALL 771
220  PRINT : INPUT "WHAT WAS IT?";G
230  PRINT : IF G = P THEN S = S + 1:A$
= "CORRECT!": GOTO 250
240 A$ = "WRONG...IT WAS " +  STR$ (P) +
 "."
250 T = T + 1: PRINT A$
260  PRINT : INPUT "KEEP TRYING?";YN$
270  IF YN$ = "Y" THEN 160
280  PRINT : PRINT "YOUR ACCURACY RATING
 IS ";S / T;".": END
290  FOR I = 771 TO 789: READ J: POKE I,
J: NEXT : RETURN
300  DATA  173,48,192,136,208,4,198,1,24
0,8,202,208,246,166,0,76,3,3,96
```



MORSE CODE

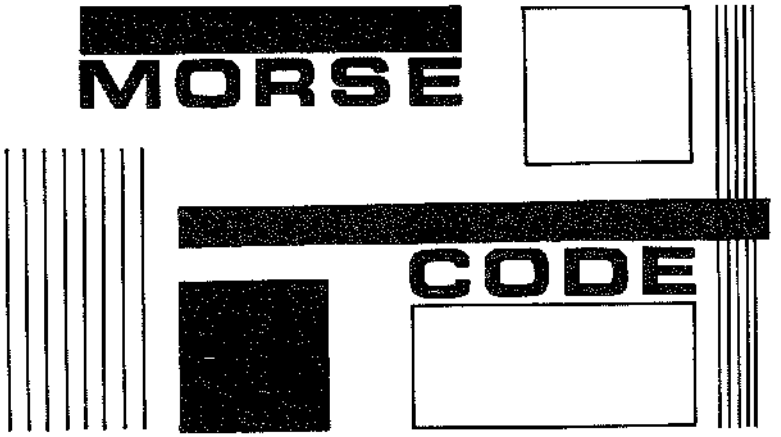
This program converts normal messages into audible Morse code, which is output through the APPLE's speaker. The message can be anything up to 255 characters in length.

Any character in the message that is not a letter will be output as a (space), which means you won't hear anything. When you type in the program, make sure you get the data absolutely perfect, or else the Morse code will be incorrect.

This program was not written for the amateur radio expert, so if you are proficient already at Morse code you may find it a little slow. If this is the case, you can change the gap between the dots and dashes by adjusting the first delay loop in line 230. You can adjust the gap between letters using the second one, or you can shorten the gap between words by altering the loop in line 160.

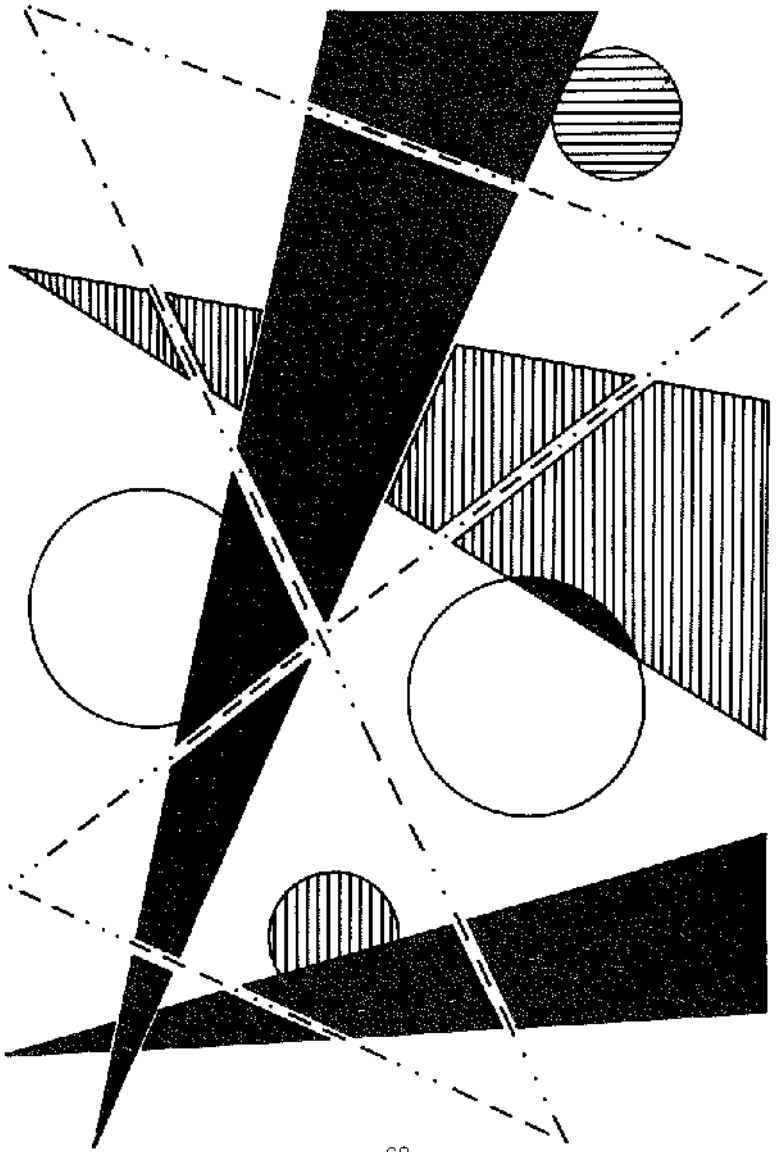
If you want to learn Morse code, putting a 'HOME' statement after the input in line 110, and getting a friend to put in a message in for you, may help.

```
10  REM  MORSE CODE
20  REM  TONY DYSON '84
30  REM  IDEA:P.BUNN,J.VINCENT
40  TEXT : HOME
50  GOSUB 250
60  GOSUB 100
70  GOSUB 130
80  PRINT : PRINT : INPUT "ANOTHER MESSA
GE? ";YN$: IF YN$ = "Y" THEN 40
90  END
100  PRINT "WHAT MESSAGE WOULD YOU LIKE
CONVERTED?"
110  PRINT : INPUT "";M$
```


```
120  RETURN
130  FOR I = 1 TO  LEN (M$)
140 A =  ASC ( MID$ ((M$),I,1)) - 64
150  IF A > 0 AND A < 27 THEN 180
160  FOR J = 1 TO 450: NEXT
170  GOTO 230
180  RESTORE : FOR J = 1 TO A: READ D$:
NEXT
190  FOR J = 1 TO  LEN (D$)
200  IF  MID$ (D$,J,1) = "." THEN  POKE
1,10: POKE 0,50: GOTO 220
210  POKE 1,70: POKE 0,100
220  CALL 771
230  FOR DL = 1 TO 30: NEXT : NEXT : FOR
 DL = 1 TO 250: NEXT : NEXT
240  RETURN
250  POKE 771,173: POKE 772,48: POKE 773
,192: POKE 774,136: POKE 775,208
260  POKE 776,4: POKE 777,198: POKE 778,
1: POKE 779,240: POKE 780,8
270  POKE 781,202: POKE 782,208: POKE 78
3,246: POKE 784,166: POKE 785,0
280  POKE 786,76: POKE 787,3: POKE 788,3
: POKE 789,96: RETURN
```

```
290  DATA  .-,-...,-.-,-..,.,..-.,--.,.
...,...,.---,-.-,.-.,--
300  DATA  -.,---,.--.,--.-,.-.,...,-,..
-,...-,.--,-..-,-.--,--..
```



HANGMAN

This is a computerised version of the old word game, Hangman. You can play with one or two players, with the second player entering the word. If you want to play with one player, you have to enter your words as data at the end of the program. Set the variable N in line 25 equal to the number of words you have entered.

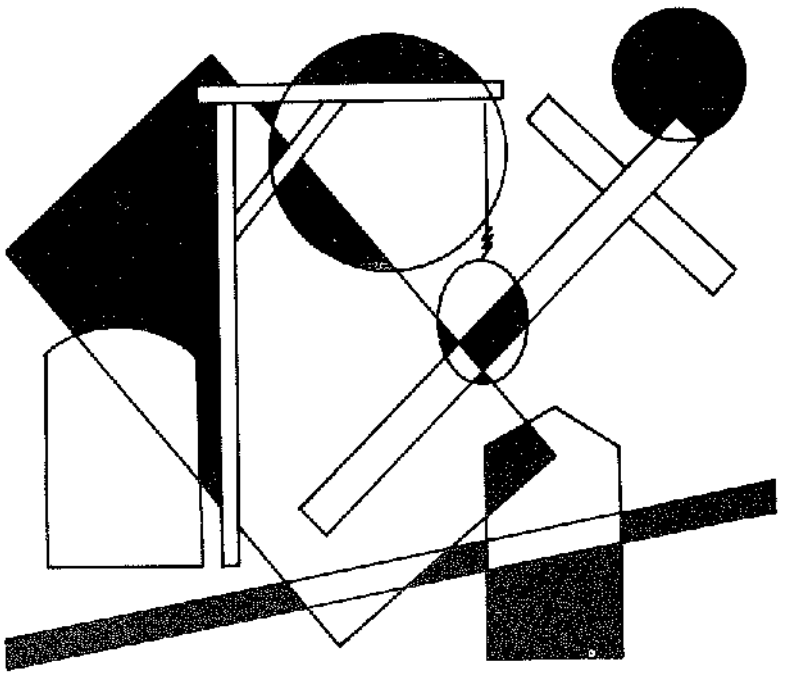
```
10  REM  HANGMAN
20  REM  TONY DYSON '83
30  GOSUB 100: REM  INITIALIZE
40  GOSUB 200: REM  CYCLE
50  GOSUB 240: REM  GET GUESS
60  GOSUB 290: REM  CHECK GUESS
70  IF ER = 1 THEN ER = 0: GOSUB 400: GO
TO 80
80  GOTO 40
90  REM  SETUP
100  HOME : HTAB 10: PRINT "**  H A N G
M A N  **"
110  VTAB 10: INPUT "HOW MANY PLAYERS (1
 OR 2) ?";P: IF P = 1 THEN 140
120  VTAB 10: CALL  - 868: INPUT "ENTER
WORD PLEASE :";A$
130  GOTO 150
140  FOR I = 1 TO  INT ( RND (1) * N) +
1: READ A$: NEXT
150 L =  LEN (A$): FOR I = 1 TO L:D$ = D
$ + "-": NEXT
160  POKE 34,20: HGR : HOME
170 T$ = "LETTERS TRIED:"
180  RETURN
190  REM  CYCLE
200  VTAB 21: PRINT T$;L$
210  VTAB 22: PRINT D$
```

```
220  RETURN
230  REM  GET GUESS
240  VTAB 23: CALL  - 868: INPUT "NEXT G
UESS :";G$
250  IF  LEN (G$) <  > 1 OR  ASC (G$) <
65 OR  ASC (G$) > 90 THEN 240
260  FOR I = 1 TO  LEN (L$): IF G$ =  MI
D$ (L$,I,1) THEN 240
270 L$ = L$ + G$: RETURN
280  REM  CHECK GUESS
290 ER = 1: FOR I = 1 TO L: IF I = 1 OR
I = L THEN 330
300 Z1$ =  RIGHT$ (D$,L - I)
310  IF G$ =  MID$ (A$,I,1) THEN D$ =  L
EFT$ (D$,I - 1) + G$ + Z1$:ER = 0
320  GOTO 370
330  IF I = 1 THEN 360
340  IF G$ =  RIGHT$ (A$,1) THEN D$ =  L
EFT$ (D$,L - 1) + G$:ER = 0
350  GOTO 370
360  IF G$ =  LEFT$ (A$,1) THEN D$ = G$
+  RIGHT$ (D$,L - 1):ER = 0
370  NEXT I: IF D$ = A$ THEN 510
380  RETURN
390  REM   GUESS WRONG
400 LW = LW + 1: ON LW GOTO 410,420,430,
440,450,460,470,480,490
410  HCOLOR= 5: HPLOT 0,110 TO 22,110: G
OTO 500
420  HPLOT 11,10 TO 11,110: GOTO 500
430  HPLOT 11,10 TO 61,10 TO 61,20: GOTO
 500
440  HCOLOR= 6: HPLOT 50,20 TO 70,20 TO
70,35 TO 50,35 TO 50,20: GOTO 500

450  HPLOT 60,35 TO 60,70: GOTO 500
460  HPLOT  TO 75,100 TO 85,100: GOTO 50
0
470  HPLOT 60,70 TO 45,100 TO 35,100: GO
TO 500
```



```
480  HPLOT 40,50 TO 60,50: GOTO 500
490  HPLOT  TO 80,50:HU = 1: GOTO 510
500  RETURN
510  IF HU = 1 THEN A$ = "YOU'RE HUNG!":
 GOTO 530
520 A$ = "YOU GOT IT!"
530  HOME : VTAB 21: CALL  - 868: PRINT
A$
540  VTAB 22: CALL  - 868: PRINT "ANOTHE
R GAME? "
550  GET YN$: IF YN$ = "Y" THEN  TEXT :
HOME : RUN
560  END
570  REM  DATA FROM HERE ON
```



SEQUENCE INTRODUCTION

This game is especially dedicated to mathematicians, although you don't have to be the world's greatest genius to play it. The computer prints out a series of numbers which follow a certain pattern — different every time. You have to try and logically work out what the pattern is, and then use your mental arithmetic powers to work out the next number in the sequence. Every time you get one wrong, the computer prints out the number you should have guessed, and you get another try.

The bigger the number, the harder it is to work it out, even if you do know the sequence.

The levels work like this:

Level 1: General formula: N = AX + B

A and B are constant, and X is equal to the previous number in the sequence.

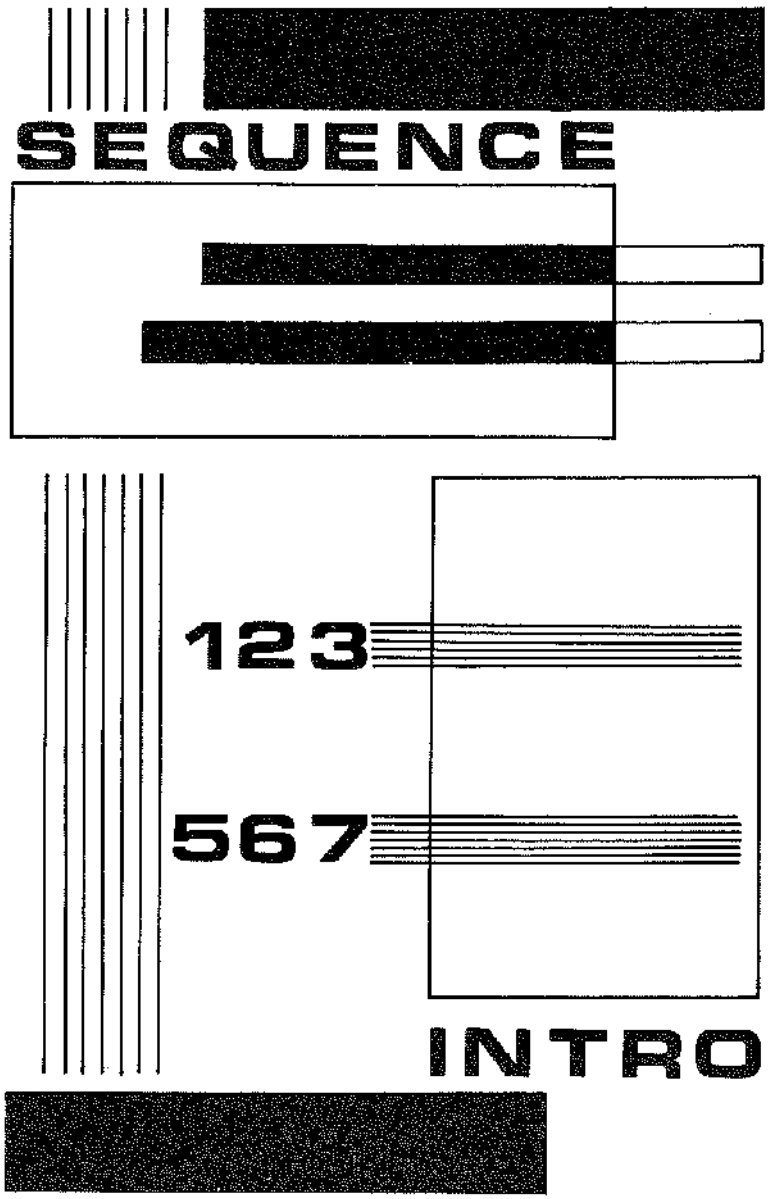
Level 2: General formula: N = AX + B

A and B can both increase or decrease by a set amount every time another number from the sequence is calculated.

For example, if A = 4 and B = 2 for the first number, A = 6 and B = −1 for the next one, A = 8 and B = −4 for the next one, and so on. The increments are randomly selected every game. As in level 1, X = the previous number in the sequence.


```
10  REM  SEQUENCE
20  REM  TONY DYSON '84
30  REM  IDEA: MY MUM
40  CLEAR : TEXT : HOME
50  VTAB 1: PRINT  TAB( 14);"-SEQUENCE-"
60  VTAB 16: PRINT "(1) MODERATE": PRINT
 "(2) HARD"
70  VTAB 14: INPUT "LEVEL: ";LV
80  DEF  FN S(N) = D * N + E
90  ON LV GOTO 130,110
100 A =  INT ( RND (1) * 11) - 5
110 DO =  INT ( RND (1) * 20) *  SGN ( R
ND (1) - 0.5)
120 EO =  INT ( RND (1) * 20) *  SGN ( R
ND (1) - 0.5)
130 D =  INT ( RND (1) * 11) - 5:DM = D
140 E =  INT ( RND (1) * 11) - 5:EM = D
150  POKE 34,1: HOME : VTAB 8
160 M = 1
170  FOR F = 1 TO 6
180  PRINT M
190 M =  FN S(M):D = D + DO:E = E + EO
200  NEXT :G = 0
210 G = G + 1: PRINT : PRINT "GUESS NO."
;G;":"
220  PRINT : INPUT A
230  IF A = M THEN 280
240  IF G = 2 * LV THEN 320
250 D = DM:E = EM
260 M = 1: FOR F = 1 TO 6 + G: PRINT M:M
 =  FN S(M):D = D + DO
270 E = E + EO: NEXT : GOTO 210
280  HOME : PRINT
290  PRINT "AT LEVEL ";LV;" YOU TOOK ";G
;" GUESSES."
300  IF G <  = LV * 10 THEN  PRINT "EXCE
LLENT! TRY LEVEL ";LV + 1;" NEXT
     TIME!"
310  END
320  PRINT "SORRY..."
```

```
330  PRINT : PRINT "BETTER LUCK NEXT TIM
E.": GOTO 310
```



DOODLE

This is a program which we have often used. It allows you to draw on the first Hi-Res screens of the Apple. You can use all the six colours available to the Apple, but due to the Apple's funny colour restrictions some colours may not be available at all places. If you own a disk you can also save and load pictures.

```
10  REM  *** DOODLE
20  REM  *** B.ENGELHARDT
30  ONERR  GOTO 100
40 D$ =  CHR$ (4)
50  REM  DOODLE
60  REM  B.ENGELHARDT
70  HOME
80  GOSUB 730: REM  INSTRUCTIONS
90 X = 10:Y = 10: GOSUB 690: REM  START
100 K =  PEEK ( - 16384)
110  HCOLOR= C
120  IF K = 206 THEN 220
130  IF K = 211 THEN O = 1: GOTO 280
140  IF K = 204 THEN O = 0: GOTO 280
150  IF K = 195 THEN 370
160  IF K = 201 THEN 440
170  IF K = 202 THEN 490
180  IF K = 203 THEN 540
190  IF K = 205 THEN 590
200  GOSUB 660
210  GOTO 100
220  POKE  - 16301,1: HOME : VTAB 24
230  GET A$
240  PRINT "ARE YOU SURE Y/N";: GET A$
250  POKE  - 16302,0
260  IF A$ = "N" THEN 100
270  GOTO 90
```

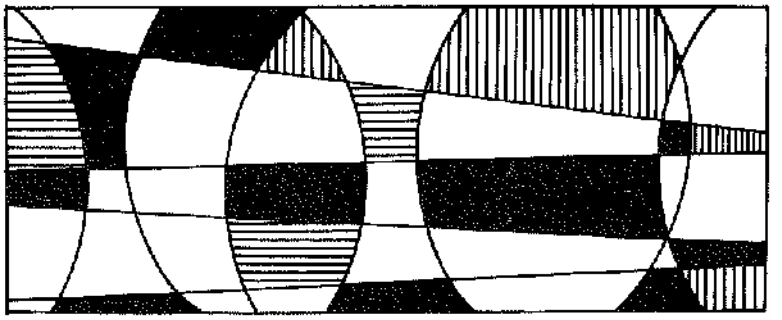
```
280  POKE  - 16301,0: HOME : VTAB 22: PR
INT "TO AND FROM DISK ONLY"
290  GET A$
300  VTAB 24: INPUT "NAME:";N$
310 D$ =  CHR$ (4)
320  IF O = 0 THEN 350
330  PRINT D$;"BSAVE ";N$;".PIC,A$2000,L
$1FFF"
340  POKE  - 16302,0: GOTO 100
350  PRINT D$;"BLOAD ";N$;",A$2000"
360  POKE  - 16302,0: GOTO 100
370  POKE  - 16301,0
380  GET A$
390  HOME
400  VTAB 22: PRINT "CURRENT COLOR=";C
410  VTAB 24: INPUT "NEW COLOR 1-6 ";C
420  HCOLOR= C: POKE  - 16302,0
430  GOTO 100
440  GOSUB 640
450 Y = Y - 1: IF Y <  = 1 THEN Y = 188
460  GOSUB 660
470  HPLOT X,Y
480  GOTO 100
490  GOSUB 640
500 X = X - 1: IF X <  = 1 THEN X = 278
510  GOSUB 660
520  HPLOT X,Y
530  GOTO 100
540  GOSUB 640
550 X = X + 1: IF X >  = 279 THEN X = 2
560  GOSUB 660
570  HPLOT X,Y
580  GOTO 100
590  GOSUB 640
600 Y = Y + 1: IF Y >  = 189 THEN Y = 2
610  GOSUB 660
620  HPLOT X,Y
630  GOTO 100
640  RETURN
650  RETURN
```



```
660  IF C = 0 THEN 680
670  HCOLOR= 0: HPLOT X,Y: FOR D = 1 TO
50: NEXT : HCOLOR= C: HPLOT X,Y: RETURN

680  HCOLOR= 3: HPLOT X,Y: FOR D = 1 TO
50: NEXT : HCOLOR= C: HPLOT X,Y: RETURN

690  HGR :C = 3: SCALE= 1: ROT= 0
700  HCOLOR= 3: HPLOT 0,0 TO 279,0 TO 27
9,191 TO 0,191 TO 0,0
710  POKE  - 16302,0
720  RETURN
730  HOME
740  PRINT : PRINT
750  HTAB 20: PRINT "I"
760  HTAB 20: PRINT "^"
770  HTAB 18: PRINT "J<+>K"
780  HTAB 20: PRINT "!"
790  HTAB 20: PRINT "M"
800  PRINT : PRINT : PRINT
810  PRINT "  N-NEW PICTURE"
820  PRINT : PRINT "  L/S-LOAD SAVE -DIS
K ONLY"
830  PRINT : PRINT "  C-COLOR-1-6"
840  PRINT : PRINT "PRESS ANY KEY TO CON
TINUE-";: GET A$: RETURN
```



How To Write Better Programs

How To Write Better Programs

INVENTING AND DEVELOPING YOUR OWN GAMES PROGRAMS

By Series Editor, Tim Hartnell

It's all very well spending your time typing in programs like those in this book, but there is sure to come a time when you decide you'd like to develop some games programs of your own. In this section of the book, I'd like to discuss a few ideas which may help you write games which you'll both enjoy developing and — more importantly — you and your friends will enjoy playing.

HAVE A CLEAR GOAL IN MIND

Although in many (perhaps most) cases, your computer program will take on a life of its own as you write it, developing away from the concept you had in mind when you started programming, it is important at the outset to have a pretty good idea of what your game will involve.

This is not as obvious a suggestion as you might think. Of course, you'll know if you're developing a 'chase the ghosts around the maze as you eat the power pills' program that you are going to need a different sort of program layout to one which places you inside a Haunted Oak, peopled with gremlins and halflings. But you have to go beyond the basic "I'm going to write me an Adventure" stage to work out such things as (a) what the object of the game will be; (b) what the screen display will look like; (c) what variables, and variable names, you'll need;

(d) the nature of the player input; (e) how 'winning' or 'losing' will be determined; and so on.

Let's look at these one by one.

THE OBJECT OF THE GAME

This can usually be stated very succinctly: "To find the lost treasure of the Aztecs"; "To destroy as many asteroids as possible before running out of ships"; or "To play a game of chess". But even though this stage of the game production can be accomplished very quickly, it should not be overlooked. Get this statement — which might be just a sentence, or may run to a paragraph length or more, if there is more than one 'screen' to be worked through, with a different scenario for each screen — down in writing.

You may well discard the original aim as the program develops, and it looks like the direction it is taking is better than the one you first thought of. Despite this, it is important to have something concrete to aim at, to stop you wasting hour after hour doodling aimlessly.

THE SCREEN DISPLAY

I've found that making a sketch, or sketches, of what the display will look like once the program is up and running, is of tremendous benefit. Once you have your drawing, and it doesn't matter how rough it is so long as it shows all the important things you want on the screen, and their relative positions and size, you'll discover the program concept is likely to crystalize.

As well as seeing immediately how you will write parts of the code to achieve the game's aim, you'll get an idea of whether or not the game is even worth writing in the form you had considered. Perhaps the game will be too complex if you leave everything on the screen you were intending to; or maybe most of the screen will be wasted space, so that a more complicated game scenario should be devised.

I've discovered that sketching the proposed screen display before starting to program is particularly useful, especially when creating arcade and simulation games. You get an indication of the variables you'll need, the user-defined graphics, the kind of player inputs which will be most conducive to good player interaction, and so on.

Simulation games, as you probably know, are those in which the computer models an external reality — such as running a cake shop, a war, or an airport — and allows you to experience (after a fashion) what it would be like to take part in such an activity in real life. Simulation games are not particularly difficult to write — in terms of needing clever coding — but instead demand a methodical, painstaking approach to the program.

In my book *The ZX Spectrum Explored* (Sinclair Browne, 1982), there is a program with the unlikely name of 'Workin' for the Man', in which you are running a factory, staffed with a highly-erratic workforce, involved in the manufacture of some mythical product called 'The Zibby'. The player gets a factory report two or three times a week, and from this report has to decide how many staff he or she will hire or (attempt to) fire, how many Zibbies will be the production target for the week, and so on.

This report is the key to the program, and when I wrote the game, I started by making a sketch of how the screen would look. It was a bit like this:

```
FACTORY REPORT: WEEK 5

Capital in hand is $2,657.92

Your stores hold 12 Zibbies worth $169.68
They sell for $14.14 each and cost $7.41
each to make

Workforce is 7 people
Their wages are $41 each and the wage bill
this week is $287
```

Each person can make 10 Zibbies a week,
a total output of 70

Once I had this sketch drawn up, I was ready to go. As you can see, it gives a very good indication of the variables which will be needed. For a start, I know I'll have to control the number of the week, the capital, the contents of the stores (and their value) and so on.

I found that once I'd completed the screen display sketch, the rest of the program was relatively easy to write. Doing a sketch in this way gives you an instant guide to the main variables you'll need.

USE HELPFUL VARIABLE NAMES

I also tend to use variable names which relate in some way to that which they are representing, as it saves having to keep a list of the variables which have been assigned, and what they've been assigned to. For example, I could use WK for week, CH for capital in hand, MZ for the cost of making each Zibby and SZ for the selling price. If Z was the number of Zibbies, I would know that the total value of Zibbies I had was Z (the number of them) multiplied by SZ (their selling price) and it cost me Z multiplied by MZ (their price of manufacture) to make them. My profit, if I sold them all, would then be Z*SZ minus Z*MZ.

If you follow a similar idea, you'll find it is much easier to keep track of what is happening in your program than might otherwise be the case.

THE NATURE OF THE PLAYER INPUT

It's important to make games *easy* and fun to play. It's not good having the best Asteroids-derivative program in the world if players have trouble hitting the fire button because you've placed it right next door to the 'rotate' control.

Many programs which provide 'up', 'down', 'right' and

'left' controls, automatically use arrow or cursor keys, even though these might be most inconvenient for the player to use. Have a look at your keyboard, and see if you can find better ones. I often use "Z" and "M" for programs which need just left and right movement, with the space bar for fire. These keys seem logical to me, and no player time is wasted in learning them, or trying to remember them when the game is underway. In a similar way, I tend to use "A" (for up) and "Z" (for down) for the left hand, and the "greater than" and "less than" keys for left and right (pointing out to the player that the < and > symbols point in the relevant directions).

Use INKEY$ or GET$ whenever you can, to prevent the player from having to use the RETURN or ENTER keys to get the program underway.

HOW THE GAME WILL END

The way the game will be won and lost needs to be defined, and clear to the player. Do you need to blast all the aliens to win, and will you lose automatically if one alien lands, and you've still got ships left, or only if you have no ships left. In a two-player game, is the loser the first player to lose three lives, or seven pieces, or does the game only end when the *difference* between the two scores is three or seven or whatever.

Work this out, and make it very clear to the player. Whether the goal of the game is to clear the left-hand side of the screen of the Screaming Widgies, or to clock up a fortune of $7.3 billion, it must be both clear to the player, and *possible to achieve*. A 'win condition' which can never be achieved on the higher levels of play is most unsatisfactory. No matter how difficult it is to do, you are only defrauding players if you set goals whose achievement is not possible within the constrictions you've put into the game.

I hope these five points may give you a few ideas on how you can go ahead and write programs which will be relatively easy to write, and which will be satisfying for you and your friends to play.

Glossary

GLOSSARY

A

Accumulator — the place within the computer in which arithmetic computations are performed and where the results of these computations are stored.

Algorithm — the series of steps the computer follows to solve a particular problem.

Alphanumeric — this term is usually used in relation to a keyboard, as in 'it is an alphanumeric keyboard', which means that the keyboard has letters as well as numbers. It is also used to refer to the 'character set' of the computer. The character set comprises the numbers and letters the computer can print on the screen.

ALU (Arithmetic/Logic Unit) — the part of the computer which does arithmetic (such as addition, subtraction) and where decisions are made.

AND — a Boolean logic operation that the computer uses in its decision-making process. It is based on Boolean algebra, a system developed by mathematician George Boole (1815-64). In Boolean algebra the variables of an expression represent a logical operation such as OR and NOR.

ASCII — stands for American Standard Code for Information Exchange, the most widely used encoding system for English language alphanumerics. There are 128 upper and lower case letters, digits and some special characters. ASCII converts the symbols and control instructions into seven-bit binary combinations.

Assembler — a program which converts other programs written in assembly language into machine code (which the computer can understand directly).

Assembly language is a low level programming language which uses easily memorised combinations of two or three letters to represent a particular instruction which the assembler then converts so the machine can understand it. Examples of these are ADD (add), and SUB (subtract). A computer programmed in assembly language tends to work more quickly than one programmed in a higher level language such as BASIC.

B

BASIC — an acronym for Beginners All-Purpose Symbolic Instruction Code. It is the most widely used computer language in the microcomputer field. Although it has been criticised by many people, it has the virtue of being very easy to learn. A great number of BASIC statements resemble ordinary English.

Baud — named after Baudot, a pioneer of telegraphic communications. Baud measures the rate of transfer of information and is approximately equal to one bit per second.

BCD — an abbreviation for Binary Coded Decimal.

Benchmark — a test against which certain functions of the computer can be measured. There are a number of so-called 'standard Benchmark tests', but generally these only test speed. This is rarely the aspect of a microcomputer that is most of interest to the potential buyer.

Binary — a numbering system that uses only zeros and ones.

Bit — an abbreviation for Binary Digit. This is the smallest unit of information a computer circuit can recognise.

Boolean Algebra — the system of algebra developed by mathematician George Boole which uses algebraic notation to express logical relationships (see AND).

Bootstrap a short program or routine which is read into

the computer when it is first turned on. It orients the computer to accept the longer, following program.

Bug — an error in a computer program which stops the program from running properly. Although it is generally used to mean only a fault or an error in a program, the term bug can also be used for a fault in the computer hardware.

Bus — a number of conductors used for transmitting signals such as data instructions, or power in and out of a computer.

Byte — a group of binary digits which make up a computer word. Eight is the most usual number of bits in a byte.

C

CAI — Computer Assisted Instruction.

CAL — Computer Assisted Learning. The term is generally used to describe programs which involve the learner with the learning process.

Chip — the general term for the entire circuit which is etched onto a small piece of silicon. The chip is, of course, at the heart of the microcomputer.

Clock — the timing device within the computer that synchronises its operations.

COBOL — a high level language derived from the words Common Business Orientated Language. COBOL is designed primarily for filing and record-keeping.

Comparator — a device which compares two things and produces a signal related to the difference between the two.

Compiler — a computer program that converts high level programming language into binary machine code so the computer can handle it.

Complement — a number which is derived from another according to specified rules.

Computer — a device with three main abilities or functions:
1) to accept data
2) to solve problems
3) to supply results

CPU — stands for Central Processing Unit. This is the heart of the computer's intelligence, where data is handled and instructions are carried out.

Cursor — a character which appears on the TV screen when the computer is operating. It shows where the next character will be printed. On a computer there are usually 'cursor control keys' to allow the user to move the cursor around the screen.

D

Data — information in a form which the computer can process.

Debug — the general term for going through a program and correcting any errors in it, that is, chasing down and removing bugs (see Bug).

Digital Computer —a computer which operates on information which is in a discrete form.

Disk/Disc — this is a magnetically sensitised plastic disk, a little smaller than a single play record. This is used for storing programs and for obtaining data. Disks are considerably faster to load than a cassette of the same length program. The disk can be searched very quickly while a program is running for additional data.

Display — the visual output of the computer, generally on a TV or monitor screen.

Dot Matrix Printer — a printer which prints either the listing of a program or that which is displayed on the TV screen. Each letter and character is made up of a number of dots. The higher the number of dots per character the finer the resolution of the printer.

Dynamic Memory — a memory unit within the computer which 'forgets' its contents when the power is turned off.

E

Editor — this term is generally used for the routine within the computer which allows you to change lines of a program while you are writing it.

EPROM — stands for Erasable Programmable Read-Only Memory. This is like the ROM in the computer, except that it is fairly easy to load material into an EPROM and it doesn't disappear when you turn the power off. EPROMs must be placed in a strong ultra violet light to erase them.

Error Messages — the information given by a computer where there is a fault in the coding during a part of a program, usually shown by the computer stopping, and printing a word, or a word and numbers, or a combination of numbers only, at the bottom of the screen. This tells you what mistake has been made. Common mistakes include using the letter O instead of zero in a line, or leaving out a pair of brackets, or one of the brackets, in an expression, or failing to define a variable.

F

File — a collection of related items of information organised in a systematic way.

Floppy Disk — a relatively cheap form of magnetic disk used for storing computer information, and so named because it is quite flexible (see Disk/Disc).

Flow Chart — a diagram drawn up before writing a program, in which the main operations are enclosed within rectangles or other shapes and connected by

lines, with arrows to represent loops, and decisions written at the branches. It makes writing a program much easier because traps such as infinite loops, or non-defined variables can be caught at an early stage. It may not be worth writing a flow chart for very short programs, but generally a flow chart aids in creating programs.

Firmware — there are three kinds of 'ware' in computers: software 'temporary' programs; hardware like the ROM which contains permanent information; and firmware in which the information is relatively permanent, as in an EPROM (see EPROM).

Flip-Flop — a circuit which maintains one electrical condition until changed to the opposite condition by an input signal.

FORTRAN — an acronym for FORmula TRANslation, this is a high level, problem orientated computer language for scientific and mathematical use.

G

Gate — an electrical circuit which, although it may accept one or more incoming signals, only sends out a single signal.

Graphics — pictorial information as opposed to letters and numbers.

H

Hard Copy — computer output which is in permanent form.

Hardware — the physical parts of the computer (also see software and firmware).

Hexadecimal (Hex) — a numbering system to the base sixteen. The digits zero to nine are used, as well as the letters A, B, C, D, E and F to represent numbers. A

equals 10, B equals 11, C equals 12, and so on. Hex is often used by microprocessor users.

Hex Pad — a keyboard designed specifically for entering hexadecimal notation.

High Level Language — a programming language which allows the user to talk to the computer more or less in English. In general, the higher the level of the language (that is, the closer it is to English), the longer it takes for the computer to translate it into a language it can use. Lower level languages are far more difficult for human operators but are generally executed far more quickly.

I

Input — the information fed into the computer via a keyboard, a microphone, a cassette or a disk.

Input/Output (I/O Device) — a device which accepts information or instructions from the outside world, relays it to the computer, and then, after processing, sends the information out in a form suitable for storing, or in a form which could be understood by a human being.

Instruction — data which directs a single step in the processing of information by the computer (also known as a command).

Integrated Circuit — a complete electronic circuit imprinted on a semiconductor surface.

Interface — the boundary between the computer and a peripheral such as a printer.

Interpreter — a program which translates the high level language fed in by the human operator, into a language which the machine can understand.

Inverter — a logic gate that changes the signal being fed in, to the opposite one.

Interactive Routine — part of a program which is

repeated over and over again until a specified condition is reached.

J

Jump Instruction — an instruction which tells the computer to go to another part of the program, when the destination of this move depends on the result of a calculation just performed.

K

K — this relates to the size of the memory. Memory is usually measured in 4K blocks. 1K contains 1,024 bytes.

Keyword — the trigger word in a line of programming, usually the first word after the line number. Keywords include STOP, PRINT and GOTO.

L

Language — computer languages are divided into three sections: high level languages, such as BASIC, which are reasonably close to English and fairly easy for humans to use; low level languages, such as Assembler, that use short phrases which have some connection with English (ADD for add and RET for return, for instance); and machine code which communicates more or less directly with the machine.

LCD — this stands for Liquid Crystal Diode. Some computers such as the TRS-80 Pocket Computer use an LCD display.

LED — this stands for Light Emitting Diode. The bright red numbers which are often used on watch or clock displays are made up of LEDs.

Logic — the mathematical form of a study of relationships between events.

Loop — a sequence of instructions within a program which is performed over and over again until a particular condition is satisfied.

M

Machine Language or Machine Code — an operation code which can be understood and acted upon directly by the computer.

Magnetic Disk — see Disk and Floppy Disk.

Mainframe — computers are generally divided into three groups, and the group a computer falls into depends more or less on its size. The computer you are thinking of buying is a microcomputer; medium sized computers are known as minicomputers; and the giant computers that you sometimes see in science fiction movies are mainframe computers. Until 15 years ago mainframe computers were, in practical terms, the only ones available.

Memory — there are two types of memory within a computer. The first is called ROM (read only memory); this is the memory that comes already programmed on the computer, which tells the computer how to make decisions and how to carry out arithmetic operations. This memory is unaffected when you turn the computer off. The second type is RAM (random access memory). This memory holds the program you type in at the keyboard or send in via a cassette or disk. In most computers the computer 'forgets' what is in RAM when you turn the power off.

Microprocessor — the heart of any computer. It requires peripheral unit interfaces, such as a power supply and input and output devices, to act as a microcomputer.

MODEM — stands for Modulator Demodulator. This is a device which allows two computers to talk to each

other over the telephone. The computers usually use a cradle in which a telephone receiver is placed.

Monitor — this has two meanings in computer terms. One meaning is a television-like display. A monitor has no facility for tuning television programs, and usually the picture produced on a monitor is superior to that produced by an ordinary television. The second meaning of a monitor relates to ROM. The monitor of a computer is described as the information it has built in when you buy it. This information allows it to make decisions and carry out arithmetic computations.

Motherboard — a framework to which extra circuits can be added. These extra circuits often give the computer facilities which are not built-in, such as that of producing sound or of controlling a light pen.

MPU — an abbreviation for Microprocessor Unit.

N

Nano-second — a nano-second is one thousand billionth of a second, the unit of speed in which a computer or a memory chip is often rated.

Non-Volatile Memory — memory which is not lost when the computer is turned off. Some of the smaller computers such as the TRS-80 Pocket Computer have non-volatile memory. The batteries hold the program you enter for several hundred hours.

Not — a Boolean logic operation that changes a binary digit into its opposite.

Null String — a string which contains no characters. It is shown in the program as two double quote marks without anything between them.

Numeric — pertaining to numbers as opposed to letters (that is, alphabetic). Many keyboards are described as being alphanumeric which means both numbers and letters are provided.

O

Octal — a numbering system which uses eight as the base, and the digits 0, 1, 2, 3, 4, 5, 6 and 7. The Octal system is not used very much nowadays in microcomputer fields. The Hexadecimal system is more common (see Hexadecimal).

Operating System — the software or firmware generally provided with the machine that allows you to run other programs.

OR — an arithmetic operation that returns a 1, if one or more inputs are 1.

Oracle — a method of sending text messages with a broadcast television signal. A teletext set is required to decode the messages. Oracle is run by Independent Television Service in the UK, and a similar service — Ceefax — is provided by the BBC.

Output — information or data fed out by the computer to such devices as a TV-like screen, a printer or a cassette tape. The output usually consists of the information which the computer has produced as a result of running a program.

Overflow — a number too large or too small for the computer to handle.

P

Pad — see Keypad.

Page — often used to refer to the amount of information needed to fill one TV screen, so you can talk about seeing a page of a program, the amount of the listing that will appear on the screen at one time.

PASCAL — a high level language.

Peripheral — anything which is hooked onto a computer, for control by the computer, such as a disk unit, a printer or a voice synthesiser.

Port — a socket through which information can be fed out of or in to a computer.

Prestel — the British telecom name for a system of calling up pages of information from a central computer via the telephone and displaying them on a television screen. A similar commercial version in the United States is known as The Source.

Program — in computer terms program has two meanings. One is the list of instructions that you feed into a computer, and the second is used as a verb, as in 'to program a computer'.

PROM — stands for Programmable Read Only Memory. This is a device which can be programmed, and once it is then the program is permanent (also see EPROM and ROM).

R

Random Access Memory (RAM) — the memory within a computer which can be changed at will by the person using the computer. The contents of RAM are usually lost when a computer is turned off. RAM is the memory device that stores the program that you type in and also stores the results of calculations in progress.

Read-Only Memory (ROM) — in contrast to RAM, information in ROM cannot be changed by the user of the computer, and the information is not lost when the computer is turned off. The data in ROM is put there by the manufacturers and tells the computer how to make decisions and how to carry out arithmetic computations. The size of ROM and RAM is given in the unit K (see K).

Recursion — the continuous repetition of a part of the program.

Register — a specific place in the memory where one or more computer words are stored during operations.

Reserved Word — a word that you cannot use for a variable in a program because the computer will read it as something else. An example is the word TO. Because TO has a specific computer meaning, most computers will reject it as a name for a variable. The same goes for words like FOR, GOTO and STOP.

Routine — this word can be used as a synonym for program, or can refer to a specific section within a program (also see Subroutine).

S

Second Generation — this has two meanings. The first applies to computers using transistors, as opposed to first generation computers which used valves. Second generation can also mean the second copy of a particular program; subsequent generations are degraded by more and more noise.

Semiconductor — a material that is usually an electrical insulator but under specific conditions can become a conductor.

Serial — information which is stored or sent in a sequence, one bit at a time.

Signal — an electrical pulse which is a conveyor of data.

Silicon Valley — the popular name given to an area in California where many semiconductor manufacturers are located.

SNOBOL — a high level language.

Software — the program which is entered into the computer by a user which tells the computer what to do.

Software Compatible — this refers to two different computers which can accept programs written for the other.

Static Memory — a non-volatile memory device which retains information so long as the power is turned on,

but does not require additional boosts of power to keep the memory in place.

Subroutine — part of a program which is often accessed many times during the execution of the main program. A subroutine ends with an instruction to go back to the line after the one which sent it to the subroutine.

T

Teletext — information transmitted in the top section of a broadcast television picture. It requires a special set to decode it to fill the screen with text information. The BBC service is known as Ceefax, the ITV service as Oracle. Teletext messages can also be transmitted by cable, for example the Prestel service in Britain or The Source in the United States.

Teletype — a device like a typewriter which can send information and also receive and print it.

Terminal — a unit independent of the central processing unit. It generally consists of a keyboard and a cathode ray display.

Time Sharing — a process by which a number of users may have access to a large computer which switches rapidly from one user to another in sequence, so each user is under the impression that he or she is the sole user of the computer at that time.

Truth Table — a mathematical table which lists all the possible results of a Boolean logic operation, showing the results you get from various combinations of inputs.

U

UHF — Ultra High Frequency (300-3000 megaHertz).

Ultra Violet Erasing — Ultra violet light must be used to erase EPROMs (see EPROM).

V

Variable — a letter or combination of letters and symbols which the computer can assign to a value or a word during the run of a program.

VDU — an abbreviation for Visual Display Unit.

Volatile — refers to memory which 'forgets' its contents when the power is turned off.

W

Word — a group of characters, or a series of binary digits, which represent a unit of information and occupy a single storage location. The computer processes a word as a single instruction.

Word-Processor — a highly intelligent typewriter which allows the typist to manipulate text, to move it around, to justify margins and to shift whole paragraphs if necessary on a screen before outputting the information onto a printer. Word-processors usually have memories, so that standard letters and the text of letters, written earlier, can be stored.

Bibliography

BIBLIOGRAPHY

Compiled by Tim Hartnell

Usborne have released a number of very attractive books in their Usborne Computer Books series. Drawing on their vast experience in the field of producing low-priced, highly-coloured, attractive books for young readers, they've produced some books which will enlighten both young and not-so-young readers.

I'll look at three of their titles, three which cover just about the whole field of computer interests:

Information Revolution
(Lynn Myring and Ian Graham, Rigby)
Presenting an eminently readable introduction to the 'revolution' which covers such fields as computers (of course), text information services via the television screen, word processing, 'future phones' and satellite communications, *Information Revolution* is an ideal guide for the person who wants an easy-to-read introduction to the field.

Computer Jargon
(Corinne Stockley and Lisa Watts).
The tone of this book is set by the frontispiece, which has a number of odd little coloured robots sitting around a table laden with computer junk, pointing at each piece saying "This is a disk drive", "This is a digital tracer" (!) and "This is a printer".

Robotics — What Robots Can Do and How They Work
(Tony Potter and Ivor Guild).
This is definitely a candidate for the award of 'the longest

title of the year'. But it is very accurate. Don't be put off by the pretty pictures, as you'll soon discover this book has a lot of solid information. Topics covered include "What robots can and cannot do", "How arm robots work", "How to teach a robot" and "Build your own micro-robot"; this last section actually includes nine pages of circuit diagrams and all to build a little two-motor robot which, following a program typed into your micro, will run about the floor. Robotics is a field of the near future (with personal robots certain to be a bigger craze — when 'real robots' finally arrive — than computers will ever be).

Practise Your BASIC
(Gaby Waters and Nick Cutler).
You'll find this book — which predictably contains a number of exercises, puzzles and problems to solve by writing programs — should be useful in giving you a number of 'core problems' which will run on your computer and which can then be modified to take advantage of your system's special features. Program listings include 'Pattern Puzzles', 'Jumping Man', 'Horse Race', 'Word Editor' and "Treasure Hunt', a mini-Adventure.

Help With Computer Literacy
(June St Clair Atkinson, Houghton Miffin).
This is a large format book with an attractive cover, fairly priced for its 122 pages. It appears to be aimed at the early to middle years of secondary education, but contains a lot of material which those teaching younger children could easily adapt. Although it avoids the 'Gee Whiz' approach of the Usborne texts, it uses cartoons and diagrams to get its message across in an inviting manner.

The Interface Computer Encyclopedia
(Ken Ozanne, Interface Publications).
Compiled by a lecturer in mathematics at the NSW Institute of Technology, this work could perhaps be more accurately called 'The Computer Book of Lists', rather

than an encyclopedia. It contains annotated references to 'all' microprocessors, 'all' microcomputers, and 'most' microcomputing magazines. The inverted commas are there because — as the author admits candidly in his introduction — any such work is likely to be out of date even before it is published. Fat (445 pages) with minimalist presentation (the whole book is dumped directly from a word processor onto a dot-matrix printer) you'll find this a useful work if you want a ready reference to chips, computers and the ever-growing field of specialist magazines.

Computer Resource Book — Algebra
(Thomas Dwyer and Margot Critchfield, Houghton Mifflin).
Dwyer and Critchfield have clocked up an enviable string of successful computer books, and this one, part of a series, shows why. With simple, but valuable programs, the authors lead the reader (who can be a secondary student, or an instructor) through most of the phrases of the BASIC programming language which are common to all low-priced computers, and most educational time-sharing systems.

Apple II BASIC
(David Goodfellow, Tab Books Inc.).
Attractively packaged, this book is clearly laid out, with an abundance of example programs; it takes a commendable approach to the business of teaching programming, with the qualities of 'programming style' introduced without fanfare. In the crowded field of 'how to program your Apple' books, this one stands out. Much of the material presented is applicable to any microcomputer.

Pre-Computer Activities
(Dorothy Diamond, Hulton Educational).
This practical guide for teachers and parents can help make children familiar with essential computer processes and language before they have hands on ex-

perience. The book contains a number of interesting activities, including investigating binary numbers using little lights, and working with cardboard 'calculators' before getting to the real thing. The discussion on computer graphics is enlivened by reference to the solid blocks which make up a 'Pacman' figure.

Word Processing Experience
(Janet Pigott and Roger Atkins-Green, Stanley Thornes Publishers Ltd.).
Designed for schools, but ideal for adapting if you'd like to increase your skill with a word processor (or simply because you'd like to see what word processors can do so you can write one for your own microcomputer), this book looks at the mechanics of word-processing, while passing on a great deal of useful information about word-processing techniques.

An Introduction to Micro-electronics and Microprocessor Systems
(G H Curtis and P G Wilks, Stanley Thornes Publishers Ltd.).
This work was written for junior college students and older school pupils, as well as for non-specialists who wanted a comprehensive — if dry — technical introduction to the subject. The going is not easy, but it's worth the effort. Topics covered include 'Logic', 'Programming the Microcomputer' and 'Analogue, Binary and Digital Systems'.

Computer Images — State of the Art
(Joseph Deken, Thames and Hudson).
This is a beautiful book, large and glossy, and packed with quality full-colour computer-generated (or, in some cases, computer-modified) images. The whole fascinating field of modern computer graphics is discussed — from television programme introductions using photographs which are colour-modified, twisted and tweeked, to the use of incredible high-resolution images in

simulators for flight training and tank manoeuvring. You'll read (and see) how computers are used to produce images, how these are used for education and communication, why 'art for art's sake' is a goal worth pursuing, and how computer images can evolve using processes uncannily akin to the processes by which groups of cells multiply and divide. If you want to see what can be done with high resolution graphics and when time, money and skill abound, you should get this book.

Computer Bluff
(Stephen Castell, Quartermaine House Ltd.).
A much more valuable book than its title indicates, it contains a lot of information on the what and how of computers, along with a generous dollop of computer jargon (or 'How to Cheat in Computer-Speak'). The style is gentle and amusing, with no appalling puns or excessive asides (such as 'didja get that joke, buster?') A pleasant, painless book which you can digest, then give to a parent.

Penguin Books has moved into the computer field with enthusiasm. As well as a 'Getting the Most Out of Your...' series, they have a number of games books. Two which stand out are **The Penguin Book of VIC 20 Games** (Paul Copeland) and **The Penguin Book of Commodore 64 Games** (Robert Young and Paul Copeland). Priced at £4.95 each, these large format books include such programs as 'Space Venture', 'Oil Rig' and 'Red Alert'. Worth buying, even if you do not have a VIC or a Commodore 64, simply as a source of ideas for new programs to create on your own microcomputer.

Arcade Games for Your VIC 20 and **Arcade Games for Your Commodore 64** (Brett Hale, Corgi/Addison-Wesley) by contrast, are definitely only for those who have the machine specified. The programs are locked irrevocably to the computer named. Taking advantage of a number of machine-specific features (such as sprite

graphics on the 64). Brett has produced a selection of around 20 programs for each machine. Each one is listed twice, the first time for the joystick and the second time for the keyboard. Titles include 'Galaxy Robbers', 'Bullet Heads' and 'Yackman'.

CREATING ADVENTURE PROGRAMS

There are a number of books, some of which are aimed at computer owners, which will help you if you are one of the many, many computer games players who are interested in developing 'Adventure' and 'Dungeons' type programs. The place to start is with TRS Hobbies' **Dungeons and Dragons** (TM) Basic Set, which comes with the introductory rule book, Dungeon Dice (tm) and an instruction module, along with a sample scenario 'The Keep on the Borderlands'. If you're new to the field, you should start with this set to give you an idea how 'real life' Adventure programs are built up.

Additional information is provided by **Fantasy Role-Playing Games** (J. Eric Holmes, Hippocrene Books Inc.) which looks at the whole field and, despite some disparaging things to say on computer versions of such games, is worth looking for. Another overview of the field — with more sympathetic comments on the use of computers — is provided by **Dicing With Dragons — An Introduction to Role-Playing Games** (Ian Livingstone, Routledge and Kegan Paul), which includes a full 'solo Adventure', a review of the major games on the market, and a fascinating chapter on the pleasures and perils of being Dungeon Master in 'Playing God'.

Fantasy Wargaming (compiled Bruce Galloway, published Patrick Stephens) provides a complete unified system for 'historically accurate' (or at least in tune with the beliefs and circumstances of individuals in the peasant, feudal-economy times in which many Adventures are set) games. The fight, weapon and

monster tables alone are worth the book, as many of their ideas can easily be incorporated into your Adventures.

There are two computer Adventure books which you could get to help you in the fascinating area of producing Adventure games on your machine.

Creating Adventure Programs on Your Computer (Andrew Nelson, Interface Publications).
Written by the author of *More Games for Your VIC 20* and *Games for Your TI 99/4A*, in the Virgin Books games series, this book takes you through the task of developing an Adventure program of your own, concentrating more on the 'Loot and Pillage' school of gaming than the Scott Adams' 'solve this puzzle to advance' field. Three complete Adventure programs are included.

Write Your Own Adventure Programs for Your Microcomputer (Jenny Tyler and Les Howarth, Usborne) is a much quicker introduction to the field than Nelson's, but nevertheless packs a lot of valuable information into its 48 pages. Step-by-step instructions are provided for creating an Adventure from scratch. A complete program — 'Haunted House' — is included in the book.

The Age of Computers is the general title of four fine books produced by Wayland Publisher Limited. Each priced at £4.95, the books present a careful, but inviting, view of four aspects of the computer field, one on the history of computers and the others looking at specific areas of modern computer application. Each book is by Ian Litterick and Chris Smithers. The four titles are **The Story of Computers,** with Charles Babbage and Uncle Sir Clive Sinclair just inside the cover (and these two pictures accurately sum up the historical period covered by the book); **How Computers Work** (with chapter headings including 'Bits, Bytes and Binary', 'Decision-making by Transistor', and 'Talking With Computers'); **Computers in Everyday Life** (such things as 'Robots in the Home', 'Magnetic Money' and 'Medicine and the Disabled');

and **Computers and You** ('Computopia', 'Big Brother', 'War and Peace' and a fascinating final chapter — 'Will Computers Need Us?').

Inside BASIC Games
(Richard Mateosian, Sybex).
This book is a slightly overwritten guide to understanding computer games. You'll learn how to write interactive programs in BASIC and how the principles of system development are applied to small computers. The book also looks at how the features of specific small computer systems have been supported in BASIC. If you can contend with the verbiage, you'll find this book well worthwhile.

1001 Things to Do With Your Personal Computer
(Mark Sawush, Tab Books).
Big and fat, and full of ideas, you'll find much here of interest to enlarge your computer horizons. The book tells you about writing music and stories with your computer, aiding a mechanic or a carpenter, solving simultaneous equations, astrology and much, much more.

Stimulating Simulations
(C.W. Engel, Hayden Book Company).
Here are 12 unique programs written in a good, general version of BASIC. The fascinating programs include 'Forest Fire', 'Rare Birds' and 'The Devil's Dungeon'. You're sure to enjoy playing those three, along with 'Diamond Thief', in which the computer decides who has committed the crime, then challenges you to discover which of the suspects is guilty. The material in this book is generally tightly programmed, and can be a helpful source of ideas to improve your own computer work.

The BASIC Handbook
(David A. Lien, Compusoft Publishing).
This is an encyclopedia of the BASIC language. It comes into its own when you find a program in a magazine or

book which you'd love to try, but are frustrated because it is written for another version of BASIC. Every BASIC word you've ever heard of (and many you may not have, such as NE, GOTO-OF and LE) is in here, along with a number of variations, one of which will almost certainly be on your machine.

BASIC Computer Games
(David Ahl, Creative Computing Press).
This is a classic work, still selling well despite the fact it was one of the first such books — if not *the* first — on the market. David Ahl has been in personal computers even before there were such things. Although several of the games are overly dependent on the random number generator, you'll find there are many, many games you'll want to adapt and improve for your own computer.

How to Buy (and Survive) Your First Computer
(Carolee Nance Kolve, McGraw-Hill Book Company).
When is a business ready for a computer? How do you make an intelligent, informed choice among the hundreds of computers available? Will a computer improve a company's operations? Answers to these and a score of similar questions are in this book, which explains in detail what to consider before buying, how to select the right computer, and what to do after ordering the computer to ensure a successful installation. Ms Kolve has over 15 years computer experience (including a stint with IBM) and brings her experience to bear in a relatively easily-digestible guide.

Your First BASIC Program
(Rodnay Zaks, Sybex).
This book, liberally illustrated with large red dinosaurs in a variety of situations vaguely related to the text (one for instance, as a cowboy getting tangled up in his ropes with the caption 'Be careful when looping'), is a gentle and worthwhile introduction to the not so-secret secrets of programming in BASIC. When you want to move

beyond just typing in other people's programs from books and magazines, this may be a good place to start.

This bibliography was compiled by the series editor, Tim Hartnell, who has felt constrained not to recommend any of his own books. However, he asked us to mention two which could be of use and interest to you.

The first is **The Personal Computer Guide** (Virgin Books) which explains what a personal computer is, and answers questions like "Will it help my kids?", "What sort of games can we play on it?" and "What can I use it for in the home?". The book describes many of the most popular computers available today, with illustrations, technical specifications and other information to help you to choose the equipment best suited to your requirements. Also included is an introduction to BASIC programming, with details of programs suitable for use in the home, a list of suppliers and user clubs, and a guide to further reading. There are also chapters covering the personal computer's history and its future. When you're ready to upgrade, you'll find this book a good, unbiased, reference work which looks at the choices facing you.

Tim Hartnell's Giant Book of Computer Games.
Described by *Personal Computer News* as 'a good source of ideas', this 386-page book, published by Fontana, for £3.95, contains over 40 programs which will run with minimum modifications on most popular microcomputers. The games include chess (of a sort!), a 17K Adventure and 'Hyperwar'.

THE VIRGIN COMPUTER GAMES SERIES

Computers are used in the home to challenge and entertain

These books meet this demand with a series of machine specific titles supplying the programs that are most sought after – those which test your skill and initiative.

The series will also improve your programming skills, as you follow the instructions to enter each of the programs into your machine.

Each title includes a concise dictionary of computer terms, a selective bibliography and hints on how to extend the programs in the book.

Each title offers computer users twenty quality programs, each one specially written for the series and unavailable elsewhere.

GAMES FOR YOUR ZX 81

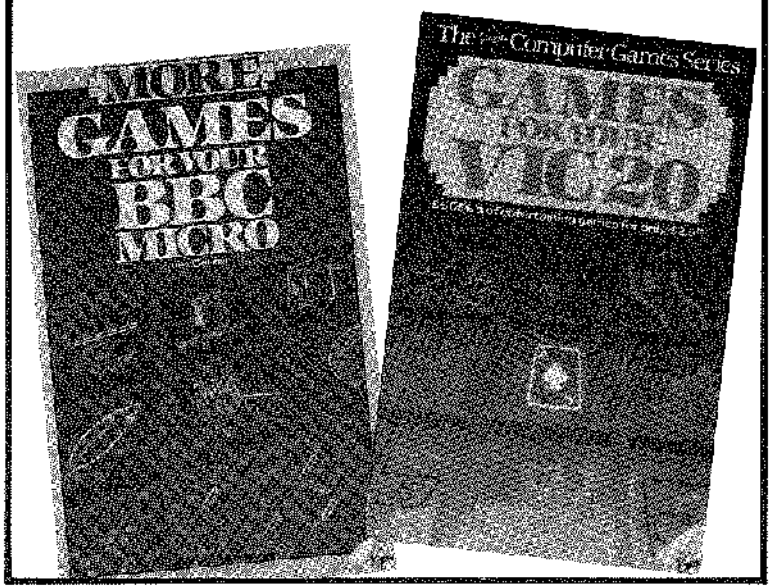
GAMES FOR YOUR ZX SPECTRUM

GAMES FOR YOUR VIC 20

MORE GAMES FOR YOUR VIC 20

GAMES FOR YOUR BBC MICRO

MORE GAMES FOR YOUR BBC MICRO



GAMES FOR YOUR DRAGON

MORE GAMES FOR YOUR DRAGON

GAMES FOR YOUR ATARI

GAMES FOR YOUR ATARI 600 XL

GAMES FOR YOUR TRS 80

GAMES FOR YOUR ORIC

MORE GAMES FOR YOUR ORIC

GAMES FOR YOUR COMMODORE 64

MORE GAMES FOR YOUR COMMODORE 64

GAMES FOR YOUR ACORN ELECTRON

GAMES FOR YOUR TI99/4A

GAMES FOR YOUR APPLE IIe

GAMES FOR YOUR SPECTRAVIDEO

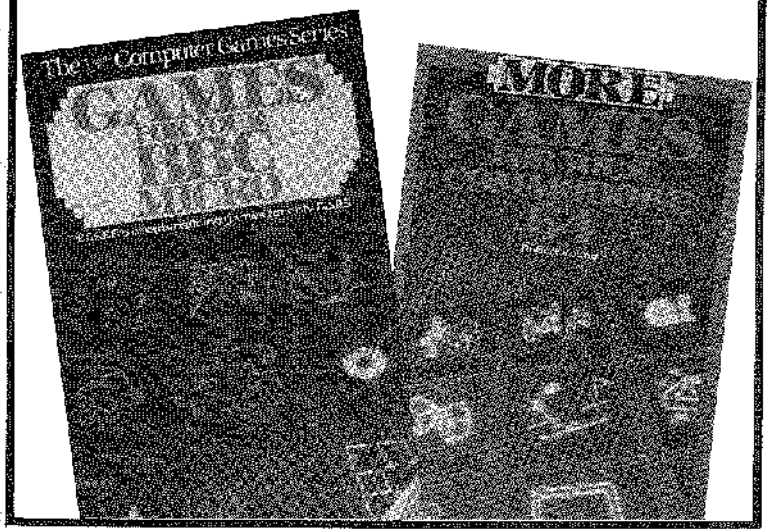
PLUS

ADVENTURES FOR YOUR COMMODORE 64

ADVENTURES FOR YOUR ZX SPECTRUM

THESE BOOKS ARE AVAILABLE AT £2.95 FROM ANY GOOD BOOKSHOP OR CAN BE OBTAINED DIRECT FROM VIRGIN BOOKS 61–63 PORTOBELLO RD, LONDON W11 3DD

PLEASE MAKE CHEQUES/ P.O.'s PAYABLE TO VIRGIN BOOKS AND INCLUDE 35p FOR POST AND PACKING.



The Virgin Computer Games Series

GAMES FOR YOUR APPLE IIe

More than 20 challenging programs, each one especially written for the series and guaranteed to provide hours of entertainment.

The great games waiting for you include PHASER (this program features real time animation: destroy the descending alien!); HOWZAT (get the County 11 cricket team to play on your TV screen); SNAKE (guide your reptile around the screen, avoiding the obstacles); SIMON SAYS (can you repeat the growing sequence of sound and colour?); CANNON (test your skill in the noble art of castle-blasting); and JACKPOT (the winnings may not be real – but the fun certainly is!).

GAMES FOR YOUR APPLE IIe will improve your programming skills as you follow the instructions to put each of the programs into your machine, and comes complete with a brief dictionary of computer terms, a selective bibliography and some hints on how to extend the programs in the book.

0 86369 046 7

Programs of originality and quality for all the family.

Virgin

United Kingdom £2.95
Australia $9.95 (recommended)